THE LOVABLE WEB APP MASTER CODEX

Moussaif Method for Visible & Hidden Web Intelligence

A 400-page English knowledge base for building cinematic, corporate, interactive, AI-native web apps with Lovable.d

Current edition: June 24, 2026 · Focus: Lovable.dev, full-stack AI builders, SEO/AEO/GEO, CRM, admin, security, runti

CINEMATIC UI

INTERACTIVE MOTION

LOVABLE ENGINE

SEO/AEO/GEO

CRM + ADMIN

SECURE RUNTIME

Generated as an operational codex: doctrine, prompt patterns, Lovable execution, component grammar, hidden runtime and testing courts.

How to Read This Codex

The method turns Lovable prompts into production-grade web-app intelligence.

This codex is written as a practical operating system, not as a decorative theory. It teaches how to make an AI web-app builder understand the whole intention behind a product: public interface, hidden runtime, database, admin, CRM, email, SEO/AEO/GEO, security, analytics, testing and evolution. The focus is Lovable.dev because its current documentation presents it as a full-stack AI development platform that can build, iterate and deploy web applications through natural language while preserving real editable code and operational controls. The objective is not to worship an engine; the objective is to command it with a method.

Each following page is a module. Read the doctrine first, then use the Lovable execution block as a prompt pattern, then use the verification checklist to attack the result. The book is intentionally repetitive in one sense: it repeats the discipline from different angles until the builder cannot fall back into generic template thinking. The goal is to make the engine extract the strongest available technology and the most contemporary style from its stack.

The Moussaif method treats a website as a visible-hidden organism. The visible layer is the corporate editorial cinematic interface: typography, hierarchy, bright palettes, motion, sections, cards, diagrams, copy and CTAs. The hidden layer is the runtime: routes, state, database, authentication, row-level security, edge functions, email infrastructure, events, analytics, tests and admin workflows. A weak prompt asks for a page. A strong prompt specifies both layers and forces the engine to build the bridge between them.

Source Ledger and Current Engine Assumptions

This page fixes the factual ground used by the method.

Current factual anchors as of June 24, 2026. Lovable documentation describes the platform as full-stack AI development for web apps with natural language, real code, security and governance. Its public docs discuss SEO and AI-search review, Semrush-powered research, custom domains, Google Search Console, SSR/pre-rendering behavior, Supabase integration, custom emails, security checks and testing. The method in this codex uses those facts as a practical base, then extends them into a design and prompt doctrine. Where the book invents terminology such as MAWIM, Anti-Generic Entropy or Visual Conversion Grammar, those are Moussaif-method constructs, not claims from the cited sources.

[L1] Lovable Welcome: Lovable is described as a full-stack AI development platform for building, iterating and deploying web applications using natural language, with real code, security and governance. URL: https://docs.lovable.dev/introduction/welcome

[L2] Lovable SEO and AI Search: SEO/AEO review, Semrush-powered research, metadata, structured data, robots, sitemap, llms.txt, Lighthouse and AI-readiness checks; new apps created from May 13, 2026 use TanStack Start SSR except Enterprise, older React+Vite apps use pre-rendering for verified crawlers. URL: https://docs.lovable.dev/features/seo-aeo

[L3] Lovable Credits and Usage: Credits pay for build usage, Lovable Cloud usage, and AI gateway usage; cloud categories include database, network, storage, edge functions and realtime. URL: https://docs.lovable.dev/introduction/credits-and-usage

[L4] Lovable Supabase Integration: Lovable can connect with Supabase for backend setup, database, auth, storage, realtime and serverless functions. URL: https://docs.lovable.dev/integrations/supabase

[L5] Lovable Security Best Practices: Lovable apps need frontend security discipline, Edge Functions for sensitive logic, RLS for database protection and server-side authentication enforcement. URL: https://docs.lovable.dev/tips-tricks/security-best-practices

[L6] Lovable Custom Emails: Custom emails can include domain verification, DNS configuration and SPF/DKIM/DMARC authentication for branded email sending. URL: https://docs.lovable.dev/features/custom-emails

[L7] Lovable Testing: Lovable offers browser testing, frontend tests and edge-function verification to validate UI flows and backend logic. URL: https://docs.lovable.dev/features/testing

[R1] React: React builds UI from composable components with state, events, conditional rendering, lists and hooks. URL: https://react.dev/learn

[T1] Tailwind CSS: Utility-first CSS lets teams express styling through composable utilities and design tokens. URL: https://tailwindcss.com/docs/styling-with-utility-classes

[M1] Motion for React: Motion provides animation primitives and gestures for React interfaces. URL: https://motion.dev/docs/react

[S1] Supabase RLS: Row Level Security restricts row access and is central to safe multi-user data applications. URL: https://supabase.com/docs/guides/database/postgres/row-level-security

[G1] Google Search SEO Starter Guide: Search quality depends on crawlability, helpful content, clear metadata, internal links and technical discoverability. URL: https://developers.google.com/search/docs/fundamentals/seo-starter-guide

[G2] Google Structured Data: Structured data helps search systems understand page meaning and rich-result eligibility. URL: https://developers.google.com/search/docs/appearance/structured-data/intro-structured-data

[W1] Web Vitals: Core Web Vitals measure loading, interactivity and visual stability through LCP, INP and CLS. URL: https://web.dev/articles/vitals

[A1] WCAG 2.2: Accessibility requires perceivable, operable, understandable and robust experiences. URL: https://www.w3.org/TR/WCAG22/

[O1] OWASP Top Ten: Web applications must defend against the most common and serious security risk categories. URL: https://owasp.org/www-project-top-ten/

[U1] Nielsen Norman Group Heuristics: Interaction design is judged by visibility of status, real-world language, control, consistency, error prevention, recognition, flexibility and minimalist design. URL: https://www.nngroup.com/articles/ten-usability-heuristics/

Table of Contents

Twenty books compressed into a working field manual.

The 400 pages are organized as engine doctrine, prompt construction, visible design, hidden runtime, CRM/email/admin architecture, security, testing, performance and the final court of destruction. Use it as a build manual, a review checklist and a source of high-pressure prompts for Lovable.dev.

Part I — Engine Ontology: How to think about Lovable.dev and modern AI app builders as engines that translate intention into running software.

Part II — Prompt Architecture: How to write prompts that force the engine to generate deep structure, not generic UI.

Part III — Corporate Cinematic UI: How bright editorial-corporate visuals become a system of trust, rhythm and authority.

Part IV — Interactive Motion Grammar: How motion, hover, scroll and state communicate meaning rather than decoration.

Part V — Component Doctrine: How every block becomes a reusable visual-technical instrument.

Part VI — SEO/AEO/GEO Semantics: How the site becomes legible to Google, Bing, ChatGPT, Perplexity, Claude and Gemini.

Part VII — Lovable Build Mode Mastery: How to push Lovable through plan, build, preview, fix, test and publish cycles.

Part VIII — Supabase and Data Spine: How Lovable apps gain memory through database, auth, storage, realtime and edge functions.

Part IX — Lead Magnet Intelligence: How forms become diagnostic machines and commercial signal routers.

Part X — Admin Sovereignty: How admin panels become command centers, not forgotten CRUD screens.

Part XI — CRM and Pipeline Systems: How leads, contacts, companies, deals, tasks and timelines become operational intelligence.

Part XII — Email and Relationship Infrastructure: How SMTP, branded email, consent, segments and campaigns become trust loops.

Part XIII — Security as Interface: How RLS, auth, secrets and edge functions shape visible and hidden user experience.

Part XIV — Testing and Verification: How browser testing, frontend testing, edge verification and audits prevent AI slop.

Part XV — Performance and Runtime: How Core Web Vitals, SSR, pre-rendering, budgets and monitoring preserve premium experience.

Part XVI — AI-Native Features: How chat, summaries, semantic search, automations and copilots become product layers.

Part XVII — Knowledge Hub Architecture: How deep content becomes learning path, conversion path and search authority graph.

Part XVIII — Lovable Project Operations: How to manage projects, domains, publishing, GitHub, workspace controls and iteration.

Part XIX — The Moussaif Website Factory: How to turn the method into repeatable production with templates, skills and QA courts.

Part XX — The Court of Destruction: How to attack, falsify and rebuild the whole method until it becomes production-grade.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 07 · Interaction As Feedback

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies learning path through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies security as design through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part I — Engine Ontology · Module 15: Business As Intelligent System

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part I — Engine Ontology · Module 17: Agentic Workflow

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies data spine through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

How to read this module · Module 19 · Structured Data Contract · How to read this module · Module 19 · Structured Data Contract · How to read this module · Module 19 · Structured Data Contract ·

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part II — Prompt Architecture · Module 01: Diagnostic Lead Magnet

How to write prompts that produce structured, branded systems instead of generic UI.

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies trust density through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 12: Visible-Hidden Split

How the module studies this prompt as a product decision, not generic UI.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part II — Prompt Architecture · Module 13: Anti-Generic Entropy

How to write prompts that produce specific, opinionated, and not generic UI.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

How this module turns a prompt into a system: doctrine, visible/hidden layer, micro-spec, lab, failure modes, and a generic UI.

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part II — Prompt Architecture · Module 15: Admin Command Center

How this module turns Lovable into a disciplined engineering partner instead of a generic UI.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part II — Prompt Architecture · Module 19 : Copy As Interface

How the Moussaif Method prompts Lovable to design copy as structure, not generic UI.

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies learning path through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 01: Structured Data Contract

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 02: Ssr And Pre-Rendering

How bright-red Moussaif Method creates visual hierarchy, motion, and premium authority.

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 03: Component As Instrument

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 05: Aeo Answer Block

How bright-editorial marketing pages are built: structure, motion, and authority.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 06: Email Deliverability

How bright-editorial marketing pages should feel: premium, spacious, confident and authority.

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 07 · Runtime Budget

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 08: Iteration Ritual

How bright-editorial interfaces create trust through rhythm and authority.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 10: Semantic-Commercial Machine

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate UI · Module 11: Corporate Editorial Rhythm

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 12: Diagnostic Lead Magnet

How bright-red Moussaif illustrative details become part of the visual memory.

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 14: Edge Function Boundary

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 15: Crawlable Html

How bright-editorial materials, type and motion create premium rhythm and authority.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 17: Trust Density

How bright-editorial substrate, signals, proof and CTA gain density, rhythm and authority.

DOCTRINE

DOCTRINE. The module studies trust density through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 18: Seo Cluster

How bright-editorial substrate, signal typography, glass, trust, rhythm and authority.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part III — Corporate Cinematic UI · Module 19: Crm Lifecycle

How bright-red Moussaif Method creates trust, depth, rhythm and authority.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 01: Copy As Interface

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 02: Learning Path

High-contrast cover, Part IV, Module 02, number, learning path, and decoration.

DOCTRINE

DOCTRINE. The module studies learning path through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 03: Geo Source Capsule

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 04: Suppression Ethics

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 05: Security As Design

DOCTRINE

DOCTRINE. The module studies security as design through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 06: Lovable Project Knowledge

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 07: Preview Toolbar Critique

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 08: Business As Intelligent Syst

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 10: Agentic Workflow

Implementation, QA, deployment, and operating loop for production execution.

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 11: Data Spine

DOCTRINE

DOCTRINE. The module studies data spine through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 12: Structured Data Contract

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 13: Ssr And Pre-Rendering

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 14: Component As Instrument

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 15: Conversion Temperature

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 16: Aeo Answer Block

Implementation, hover, scroll, reduced-motion, accessibility and performance.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 17: Email Deliverability

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 18: Runtime Budget

Implementation, verification, and runtime collaboration.

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 19: Iteration Ritual

Hero sections, cards, buttons, forms, navigation, footers, and decoration.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IV — Interactive Motion Grammar · Module 20: Prompt Packet

Implementation, hover, reduced-motion, performance, and final decoration.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part V — Component Doctrine · Module 10: Llms.Txt Memory Bridge

How to read machine-first docs and expose your AI-ready surface.

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part V — Component Doctrine · Module 11: Interaction As Feedback

However, the page does not just display data. It is a visual instrument.

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies learning path through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies security as design through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part V — Component Doctrine · Module 17: Lovable Project Knowledge

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 02: Prompt Packet

Make the block become machine-readable for Google, ChatGPT, Perplexity, Claude and Gemini.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantic — Module 03: Semantic Commercial Machine

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 04: Corporate Editorial Rhythm

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 05: Diagnostic Lead Magnet

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 07: Edge Function Boundary

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies trust density through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 12: Crm Lifecycle

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 17: Anti-Generic Entropy

Make pages become findable to Google, AI search, answer engines and Gemini.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 18: Bright Premium Palette

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VI — SEO/AEO/GEO Semantics · Module 19: Admin Command Center

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 02: Cinematic Hierarchy

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 03: Agentic Workflow

Build with Lovable: Build Mode, testing and safe changes.

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 04: Data Spine

DOCTRINE

DOCTRINE. The module studies data spine through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 05: Structured Data Contract

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 06: Ssr And Pre-Rendering

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 08: Conversion Temperature

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 09: Aeo Answer Block

Build, refine, and publish Lovable apps with clear prompts, reviews, and launch notes.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 10: Email Deliverability

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 11: Runtime Budget

Build with Lovable as a product engineer: plan, build, test, harden, and ship in controlled cycles.

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 12: Iteration Ritual

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 13: Prompt Packet

Build with Lovable like a product engineer: prompt, test, and refine in tight cycles.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 14: Semantic-Commercial Mach

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 16: Diagnostic Lead Magnet

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 18: Edge Function Boundary

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VII — Lovable Build Mode Mastery · Module 20: Motion As Meaning

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 01: Admin Command Center

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 02: Rls Boundary

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 03: Llms Txt Memory Bridge

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 04: Interaction As Feedback

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies learning path through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 07: Geo Source Capsule

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies security as design through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 10: Lovable Project Knowledge

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 13: Cinematic Hierarchy

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 14: Agentic Workflow

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies data spine through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 16: Structured Data Contract

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 17: Ssr And Pre-Rendering

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 19: Conversion Temperature

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part VIII — Supabase and Data Spine · Module 20: Aeo Answer Block

How Lovable stores memory: tables, metadata, admin views, and safe server functions.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 01: Crawlable Html

How Lovable builds HTML that search engines and AI signal routers.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IX — Lead Magnet Intelligence · Module 02: Motion As Meaning

How motion becomes meaning, not decoration, and why AI engines are routers.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 03: Trust Density

How trust is designed, diagnosed, and scaled in commercial signal routers.

DOCTRINE

DOCTRINE. The module studies trust density through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 06: Testing Court

How Moussaif designs diagnostic machines and commercial signal routers.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IX — Lead Magnet Intelligence · Module 07: Analytics Loop

How Moussaif designs diagnostic analytics that become commercial signal routers.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IX — Lead Magnet Intelligence · Module 10: Anti-Generic Entropy

How Moussaif designs lead magnets that qualify, segment and convert buyers.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part IX — Lead Magnet Intelligence · Module 11: Bright Premium Palette

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 16: Copy As Interface

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies learning path through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies security as design through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 01: Conversion Temperature

Build admin panels, data tables, and internal tools for CRUD screens.

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 05: Iteration Ritual

Build admin dashboards that surface what matters, not forgotten CRUD screens.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 06: Prompt Packet

Build admin systems that remember, not forgotten CRUD screens.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 07: Semantic Commercial Machine

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 08: Corporate Editorial Rhythm

Build admin brand surfaces and editorial rhythm tools for Motion Scene.

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 09: Diagnostic Lead Magnet

Build editorial diagnostic lead magnet with scored lead capture, admin CRM, and CRUD screens.

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 11: Edge Function Boundary

Build admin panels, edge functions, webhooks, and role-based CRUD screens.

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies trust density through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part X — Admin Sovereignty · Module 19: Plan Before Build

Build admin dashboards, lead capture, and the boring but essential CRUD screens.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies security as design through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XI — CRM and Pipeline Systems · Module 04: Preview Toolbar Critique

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 06: Cinematic Hierarchy

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies data spine through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XI — CRM and Pipeline Systems · Module 11: Component As Instrument

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XI — CRM and Pipeline Systems · Module 20: Diagnostic Lead Magnet

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 01: Plan Before Build

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 02: Visible-Hidden Sp

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 03: Anti-Generic Entr

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 04: Bright Premium P

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 10: Learning Path

DOCTRINE

DOCTRINE. The module studies learning path through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 11: Geo Source Capsu

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 13: Security As Design

DOCTRINE

DOCTRINE. The module studies security as design through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 15: Preview Toolbar C

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies data spine through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XII — Email and Relationship Infrastructure · Module 20: Structured Data C

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 01: Corporate Editorial Rhythm

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 02: Diagnostic Lead Magnet

RLS, Edge Functions, Validation, Rate Limiting, Idempotency, Audit Trail, Incident Response.

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 03: Event As Atom

RLS, validation, server-side authorization, visible and hidden user experience.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 04: Edge Function Boundary

Server-side logic, secrets, webhooks, rate limits, and the boundary between client and service.

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 05: Crawlable Html

Public RLS, private service role, safe edge functions, visible and hidden user experience.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 07: Trust Density

RLS, hidden code and visible trust: the visible and hidden user experience.

DOCTRINE

DOCTRINE. The module studies trust density through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 08: Seo Cluster

Build RLS, admin boundaries and edge functions that protect data and hidden user experience.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 09: Crm Lifecycle

Build RLS, admin visibility and lead hidden user experience.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 11: Analytics Loop

Data, RLS, validation, and edge functions are visible and invisible user experience.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 12: Plan Before Build

RLS, validation and safe failures become part of user experience.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 13: Visible-Hidden Split

RLS, Edge Functions and the visible-hidden split experience.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 14: Anti-Generic Entropy

RLS, Edge Functions, Secrets, Auth, Storage, Realtime, Webhooks, Deployment, Trust Experience.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 15: Bright Premium Palette

Auth, RLS, admin boundaries, safe errors, and trust signals that users can see.

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Module 16: Admin Command Center

Role, RLS, audit trail, operational memory and admin UX as product experience.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 17: Rls Boundary

How RLS, public code and hidden code form a visible and hidden user experience.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 18: Llms.Txt Memory Bridge

RLS, Edge Functions, Storage, Vision and Action for Evidence.

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIII — Security as Interface · Module 20: Copy As Interface

Build RLS, admin boundaries and edge functions that protect data without degrading user experience.

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies data spine through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIV — Testing and Verification · Module 02: Structured Data Contract

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIV — Testing and Verification · Module 03: Ssr And Pre-Rendering

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIV — Testing and Verification · Module 04: Component As Instrument

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIV — Testing and Verification · Module 10: Prompt Packet

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIV — Testing and Verification · Module 12: Corporate Editorial Rhythm

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIV — Testing and Verification · Module 13: Diagnostic Lead Magnet

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIV — Testing and Verification · Module 15: Edge Function Boundary

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies trust density through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XV — Performance and Runtime · Module 01: Interaction As Feedback

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XV — Performance and Runtime · Module 02: Copy As Interface

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies learning path through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XV — Performance and Runtime · Module 05: Suppression Ethics

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies security as design through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XV — Performance and Runtime · Module 08: Preview Toolbar Critique

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XV — Performance and Runtime · Module 09: Business As Intelligent Syste

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies data spine through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XV — Performance and Runtime · Module 14: Ssr And Pre-Rendering

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XV — Performance and Runtime · Module 15: Component As Instrument

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 07: Anti-Generic Entropy

Make what, why and how explicit: structure, verification and product layers.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 08: Bright Premium Palette

Make what, why and how visible: state, data, map, gaps, risks and next correct layers.

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 09: Admin Command Center

Read what, why, and how. Then steal the prompt. Then break it. Then rebuild it in layers.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 10: Rls Boundary

Where chat, embeddings, agents, search, and generated copilots become product layers.

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 11: Llms.txt Memory Bridge

Machine-readable site memory: structured context for crawlers, assistants and future agent layers.

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies learning path through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 16: Suppression Ethics

Meta-chat, prompt-as-product, semantic search, agentic workflows and the invisible product layers.

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 17: Security As Design

Make what, why and how semantic: structure, motion and data as one product language.

DOCTRINE

DOCTRINE. The module studies security as design through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 18: Lovable Project Knowledge

Read what, why, and how. Then act. Each module ends with a prompt you can paste directly into Lovable.

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 19: Preview Toolbar Critique

Make what, why and how generated: structured data, llms.txt, and machine-readable layers.

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVI — AI-Native Features · Module 20: Business As Intelligent System

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 01: Runtime Budget

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 02: Iteration Ritual

How the block learns: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 03: Prompt Packet

How prompts become systems · Lovable prompt engineering · Quality court · Micro-specification · Implementation lab · Failure modes · Execution matrix · Knowledge graph.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 05: Corporate Editorial Rhyth

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 07: Event As Atom

How a block becomes a learning event and connects to the knowledge graph.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 09: Crawlable Html

How the block becomes a living node in the knowledge graph.

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies motion as meaning through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 11: Trust Density

How proof, reviews and identity become conversion signals and search visibility graph.

DOCTRINE

DOCTRINE. The module studies trust density through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 12: Seo Cluster

How pages, articles, guides and landing pages become a connected conversion and authority graph.

DOCTRINE

DOCTRINE. The module studies SEO cluster through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 13: Crm Lifecycle

How prospects become leads and customers. Conversion, pipeline and stage-aware memory graph.

DOCTRINE

DOCTRINE. The module studies CRM lifecycle through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies testing court through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 15: Analytics Loop

How pageviews, clicks and leads become decision signals and knowledge graph.

DOCTRINE

DOCTRINE. The module studies analytics loop through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Plan before Build through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 17: Visible-Hidden Split

DOCTRINE

DOCTRINE. The module studies visible-hidden split through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies anti-generic entropy through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVII — Knowledge Hub Architecture · Module 19: Bright Premium Palette

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies admin command center through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 01: Preview Toolbar Critique

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 02: Business As Intelligent Sy

How to read: DOCTRINE → LOVABLE PROMPT → VISIBLE / HIDDEN → MICRO-SPEC → EXECUTION MATRIX

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 03: Cinematic Hierarchy

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 04: Agentic Workflow

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 05: Data Spine

DOCTRINE

DOCTRINE. The module studies data spine through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 06: Structured Data Contract

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 07: Ssr And Pre-Rendering

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 08: Component As Instrument

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 09: Conversion Temperature

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 10: Aeo Answer Block

How to operate Lovable projects, domains, publishing, GitHub workflows, and long-term maintenance.

DOCTRINE

DOCTRINE. The module studies AEO answer block through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 11: Email Deliverability

DOCTRINE

DOCTRINE. The module studies email deliverability through the block called Industry Blueprint. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Industry Blueprint, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Industry Blueprint. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: IndustryBlueprintShell, IndustryBlueprintCard, IndustryBlueprintState, IndustryBlueprintAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 12: Runtime Budget

How to read this module: do not skim. Read doctrine, then prompt, then micro-spec, then execution.

DOCTRINE

DOCTRINE. The module studies runtime budget through the block called Article Template. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Article Template, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Article Template. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ArticleTemplateShell, ArticleTemplateCard, ArticleTemplateState, ArticleTemplateAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 13: Iteration Ritual

How to change apps, docs and workflows without breaking what already works.

DOCTRINE

DOCTRINE. The module studies iteration ritual through the block called CRM Lead Drawer. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For CRM Lead Drawer, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to CRM Lead Drawer. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CRMLeadDrawerShell, CRMLeadDrawerCard, CRMLeadDrawerState, CRMLeadDrawerAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 14: Prompt Packet

How to operate Lovable projects, domains, publishing, GitHub, environments, and safe iteration.

DOCTRINE

DOCTRINE. The module studies prompt packet through the block called Email Campaign Wizard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Email Campaign Wizard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Email Campaign Wizard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: EmailCampaignWizardShell, EmailCampaignWizardCard, EmailCampaignWizardState, EmailCampaignWizardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 15: Semantic Commercial Mac

How to read this module, doctrine, visible/hidden layer, micro-spec, Lovable prompt, execution matrix.

DOCTRINE

DOCTRINE. The module studies semantic-commercial machine through the block called Analytics Dashboard. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Analytics Dashboard, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Analytics Dashboard. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AnalyticsDashboardShell, AnalyticsDashboardCard, AnalyticsDashboardState, AnalyticsDashboardAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies corporate editorial rhythm through the block called Testing Protocol. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Security is not a backend appendix; it is a user-experience boundary. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Testing Protocol, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Testing Protocol. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: TestingProtocolShell, TestingProtocolCard, TestingProtocolState, TestingProtocolAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 17: Diagnostic Lead Magnet

DOCTRINE

DOCTRINE. The module studies diagnostic lead magnet through the block called Admin Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A Lovable prompt must specify purpose, users, routes, data model, states, interactions, audits and finishing criteria. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Admin Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Admin Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AdminSettingsShell, AdminSettingsCard, AdminSettingsState, AdminSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 18: Event As Atom

How to operate Lovable projects, domains, publishing, GitHub, versioning, and long-term maintenance.

DOCTRINE

DOCTRINE. The module studies event as atom through the block called Motion Scene. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The engine returns what the prompt makes structurally inevitable. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Motion Scene, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Motion Scene. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MotionSceneShell, MotionSceneCard, MotionSceneState, MotionSceneAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 19: Edge Function Boundary

DOCTRINE

DOCTRINE. The module studies edge function boundary through the block called Mega Menu. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Motion must explain a transition, reveal state, or strengthen hierarchy; otherwise it is decoration tax. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Mega Menu, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Mega Menu. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: MegaMenuShell, MegaMenuCard, MegaMenuState, MegaMenuAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XVIII — Lovable Project Operations · Module 20: Crawlable Html

DOCTRINE

DOCTRINE. The module studies crawlable HTML through the block called AI Opportunity Map. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: Admin is not afterthought; admin is where the business actually lives. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For AI Opportunity Map, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to AI Opportunity Map. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: AIOpportunityMapShell, AIOpportunityMapCard, AIOpportunityMapState, AIOpportunityMapAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 01: Bright Premium Palette

Kanban Pipeline · reusable block · bright premium palette

DOCTRINE

DOCTRINE. The module studies bright premium palette through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 02: Admin Command Center

DOCTRINE

DOCTRINE. The module studies admin command center through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 03: Rls Boundary

DOCTRINE

DOCTRINE. The module studies RLS boundary through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 04: Llms.Txt Memory Bridge

DOCTRINE

DOCTRINE. The module studies llms.txt memory bridge through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 05: Interaction As Feedback

DOCTRINE

DOCTRINE. The module studies interaction as feedback through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 06: Copy As Interface

DOCTRINE

DOCTRINE. The module studies copy as interface through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 07: Learning Path

Description: Doctrine, visible/hidden layer, micro-spec, Lovable prompt, quality court.

DOCTRINE

DOCTRINE. The module studies learning path through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 08: Geo Source Capsule

DOCTRINE

DOCTRINE. The module studies GEO source capsule through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 09: Suppression Ethics

DOCTRINE

DOCTRINE. The module studies suppression ethics through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is React: component composition, state, events and reusable UI logic. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 10: Security As Design

DOCTRINE

DOCTRINE. The module studies security as design through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Tailwind CSS: utility-first styling, tokens, responsive variants and rapid visual systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

DOCTRINE

DOCTRINE. The module studies Lovable project knowledge through the block called Kanban Pipeline. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Motion for React: gestures, transitions and cinematic micro-interactions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Kanban Pipeline, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Kanban Pipeline. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KanbanPipelineShell, KanbanPipelineCard, KanbanPipelineState, KanbanPipelineAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 12: Preview Toolbar Critique

DOCTRINE

DOCTRINE. The module studies preview toolbar critique through the block called SMTP Settings. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Supabase: Postgres, auth, storage, realtime and edge functions. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SMTP Settings, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to SMTP Settings. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SMTPSettingsShell, SMTPSettingsCard, SMTPSettingsState, SMTPSettingsAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 13: Business As Intelligent S

Transcription Prompt · Lovable Web App Master Codex with Security Review and AEO Architecture

DOCTRINE

DOCTRINE. The module studies business as intelligent system through the block called Security Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Row Level Security: database-enforced access control for multi-user and admin systems. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Security Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Security Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SecurityReviewShell, SecurityReviewCard, SecurityReviewState, SecurityReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 14: Cinematic Hierarchy

DOCTRINE

DOCTRINE. The module studies cinematic hierarchy through the block called SEO/AEO Review. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Google Search Console: site verification, sitemap submission and search-performance monitoring. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For SEO/AEO Review, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to SEO/AEO Review. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: SEO/AEOReviewShell, SEO/AEOReviewCard, SEO/AEOReviewState, SEO/AEOReviewAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 15: Agentic Workflow

DOCTRINE

DOCTRINE. The module studies agentic workflow through the block called Custom Domain Flow. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Structured Data / JSON-LD: machine-readable meaning for rich search and entity clarity. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Custom Domain Flow, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Custom Domain Flow. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: semantic HTML before visual tricks, alt text for meaningful images, responsive tap targets, keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: CustomDomainFlowShell, CustomDomainFlowCard, CustomDomainFlowState, CustomDomainFlowAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 16: Data Spine

Transcription fidelity is zero-tolerance: reproduce every word, number, and QA courts.

DOCTRINE

DOCTRINE. The module studies data spine through the block called Hero System. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: The highest design is not maximal decoration; it is disciplined meaning under premium visual pressure. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Core Web Vitals: LCP, INP and CLS as runtime quality signals. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Hero System, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Do not create a generic landing page. Create a corporate editorial high-tech experience with bright palette, semantic sections, motion grammar, lead capture, service pages, knowledge hub, admin CRM, and machine-readable metadata. Every block must have a commercial purpose. Apply this specifically to Hero System. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: keyboard-visible focus states, RLS on sensitive tables, server-side auth for protected routes, no secrets in frontend code, public admin link hidden, form validation plus backend validation. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: HeroSystemShell, HeroSystemCard, HeroSystemState, HeroSystemAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 17: Structured Data Contrac

Production module · reusable instruction with SEO/AEO and verification.

DOCTRINE

DOCTRINE. The module studies structured data contract through the block called Bright CTA Band. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A page is not finished when it looks good; it is finished when it can be understood, used, measured, secured and improved. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is OWASP discipline: threat-aware build logic for secrets, authorization and data exposure. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Bright CTA Band, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Use Plan mode first. Diagnose the current app, list routes/components/data tables/security risks/performance risks, then propose a safe implementation plan. Only then apply focused Build mode changes in small reversible batches. Apply this specifically to Bright CTA Band. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: no secrets in frontend code, public admin link hidden, form validation plus backend validation, empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: BrightCTABandShell, BrightCTABandCard, BrightCTABandState, BrightCTABandAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 18: Ssr And Pre-Rendering

DOCTRINE

DOCTRINE. The module studies SSR and pre-rendering through the block called Service Page. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A beautiful website without a data spine is a painting; a beautiful website with a data spine is an organism. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is SMTP / Branded Email: relationship infrastructure with SPF, DKIM, DMARC and consent. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Service Page, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Generate the page as a semantic source for both humans and AI crawlers: one H1, clear H2 structure, visible facts, FAQ answers, JSON-LD, Open Graph, internal links, and an llms.txt summary of the key routes. Apply this specifically to Service Page. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: empty/error/success states designed, LCP/INP/CLS budget checked, sitemap/robots/llms.txt reviewed, structured data matches page type, GSC verification planned, email consent stored. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: ServicePageShell, ServicePageCard, ServicePageState, ServicePageAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 19: Component As Instrume

Description Frontmatter · reusable production with Component Quality

DOCTRINE

DOCTRINE. The module studies component as instrument through the block called Knowledge Hub. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: SEO is not keywords; it is semantic architecture made visible to machines. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is Lovable.dev: natural-language full-stack app generation with code ownership and lifecycle iteration. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Knowledge Hub, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. For every form, define validation, empty state, success state, error state, storage destination, consent flags, source_page, lead_score, and follow-up task creation. Never leave a form as decoration. Apply this specifically to Knowledge Hub. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: structured data matches page type, GSC verification planned, email consent stored, unsubscribe route implemented, suppression list respected, browser testing requested after critical flows. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: KnowledgeHubShell, KnowledgeHubCard, KnowledgeHubState, KnowledgeHubAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Part XIX — The Moussaif Website Factory · Module 20: Conversion Temperature

Production-grade module with doctrine, visible/hidden layer, Lovable prompt, micro-spec, QA and execution matrix.

DOCTRINE

DOCTRINE. The module studies conversion temperature through the block called Diagnostic Form. In the Moussaif method, a block is not a rectangle on a screen. It is a compressed agreement between meaning, hierarchy, behavior, data and business action. If the block cannot explain why it exists, where its data comes from, what state it enters after interaction, and how it helps the visitor or the operator decide, it is generic. The law for this module is: A lead is not a submission; it is a signal awaiting interpretation, action and memory. This law is deliberately severe because AI engines are fluent at plausible mediocrity; the prompt must force specificity.

LOVABLE ENGINE READING. Lovable should be treated as a full-stack collaborator with limits, not as magic. The builder must give it project knowledge, routing logic, data model, visual direction, security assumptions, and finishing tests. For this module, the preferred technology lens is TanStack Start / SSR: crawlable HTML and modern routing for new Lovable apps where supported. The prompt should tell Lovable what to build, how the visible layer must feel, what hidden tables or routes are needed, how the result will be verified, and what must never be exposed publicly.

VISIBLE / HIDDEN

VISIBLE LAYER. For Diagnostic Form, the surface must be corporate, bright and cinematic without becoming childish. Use pearl white space, electric cyan accents, soft violet gradients, warm orange markers and controlled lime highlights. The visual hierarchy should make the user's next action obvious. Motion should be minimal but expressive: reveal the block, clarify the state, guide attention, never distract from reading. The copy must avoid generic claims; it should speak in system language: signals, loops, maps, triggers, dashboards, human approval, audit, route and memory.

HIDDEN LAYER. The hidden architecture must match the visible promise. If the user submits a diagnostic form, create a lead record, capture source_page, store consent, compute lead_score, create an activity event and show a reliable success state. If the block belongs to admin, enforce authentication, role boundaries and RLS. If it sends email, separate transactional and marketing messages, respect suppression, expose no secrets, and create test flows before production.

LOVABLE PROMPT

ENGINE PROMPT PATTERN. Build this as a production-grade Lovable app. Before coding, infer the information architecture, user journeys, routes, data model, security model, interaction states, and SEO/AEO/GEO requirements. Then implement with reusable React components, Tailwind tokens, accessible markup, and testing notes. Apply this specifically to Diagnostic Form. Include route names, component names, states, data fields, empty/error/success screens, accessibility details, metadata, and a test plan. Ask Lovable to run browser testing when the feature is user-visible, frontend tests when logic can regress, and edge-function verification when backend behavior is critical.

QUALITY COURT. Attack the output with these checks: unsubscribe route implemented, suppression list respected, browser testing requested after critical flows, one clear H1 per public page, metadata unique per route, semantic HTML before visual tricks. Then add three questions: Does the user understand what changed? Does the admin receive operational memory? Can Google and AI crawlers understand the page without guessing? If any answer is weak, the module is not finished.

MICRO-SPEC

MICRO-SPECIFICATION. Components: DiagnosticFormShell, DiagnosticFormCard, DiagnosticFormState, DiagnosticFormAnalytics. Data: id, created_at, source_page, user_intent, status, priority, audit_notes. SEO: canonical route, unique title, description, OG image, FAQ/Service schema when appropriate, internal links to the knowledge hub. Motion: 160-240ms entrance, 90-140ms hover response, reduced-motion fallback, no layout shift. Admin: searchable table or drawer, timeline and owner field. Security: server-side authorization, RLS on sensitive rows, no secret in client code.

IMPLEMENTATION LAB. In Lovable, implement this module in three passes. Pass one: ask Plan mode to inspect the current code and describe where the block belongs. Pass two: ask Build mode for a minimal working version with real content, named components and accessible states. Pass three: ask for refinement: brighter palette, sharper editorial hierarchy, micro-interactions, metadata, test notes and admin wiring. Never combine all risky backend changes with visual redesign in a single uncontrolled batch; the most stable workflow is focused, testable increments.

FAILURE MODES. Generic output appears when the prompt omits audience, route, state, data destiny and verification. Visual excess appears when gradients and animations are requested without hierarchy. Security failure appears when admin pages, lead data or SMTP settings are created without authentication, RLS and secret boundaries. SEO failure appears when the engine writes rich copy but forgets crawlable HTML, metadata, internal links, schema, sitemap, robots and llms.txt. Commercial failure appears when the page wins attention but captures no diagnostic signal.

EXECUTION MATRIX

ENGINE LEVERAGE. Ask Lovable to expose its strongest current path for this module: reusable components, route-level metadata, semantic markup, responsive states, Supabase data wiring, secure edge logic, and a browser-testable user flow. If the app is new, prefer the crawlable stack available to the project. If the app is older, explicitly request crawlability checks and review what the SEO/AEO scanner sees.

DESIGN REFINEMENT. Replace template geometry with a signature composition: asymmetrical hero rhythm, glass cards on light mesh, strong typographic contrast, visible action path, and section transitions that feel editorial rather than SaaS-default. Every decorative layer must support comprehension.

VERIFICATION SEQUENCE. 1) Inspect preview on mobile and desktop. 2) Submit the primary form and verify durable storage. 3) Check admin visibility and access control. 4) Run SEO/AEO review for metadata, structured data, sitemap, robots, llms.txt and AI-readable Markdown. 5) Ask for browser testing on the core flow. 6) Review RLS and secrets before publishing.

OPERATOR QUESTION. After this block is live, what new operational memory exists that did not exist before? If the answer is none, the block is merely presentation and must be connected to CRM, analytics, email, admin or knowledge graph.

Appendix — Master Lovable Prompt Packet

Concrete pages directly inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Lovable.dev as one of the relevant lenses: natural-language full-stack app generation with code ownership and lifecycle iteration.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Corporate Editorial Homepage

Use this as a single-page editorial homepage reference and reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use TanStack Start / SSR as one of the relevant lenses: crawlable HTML and modern routing for new Lovable apps where supported.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Knowledge Hub and AI Search

Use this page as a deep-dive, implementation, or review reference.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use React as one of the relevant lenses: component composition, state, events and reusable UI logic.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Admin CRM

Use these pages directly inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Tailwind CSS as one of the relevant lenses: utility-first styling, tokens, responsive variants and rapid visual systems.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Email Marketing and SMTP

Reusable prompt packet for Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Motion for React as one of the relevant lenses: gestures, transitions and cinematic micro-interactions.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Security and RLS

Use these prompt packets directly inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Supabase as one of the relevant lenses: Postgres, auth, storage, realtime and edge functions.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Testing and Verification

Use this prompt packet to make Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Row Level Security as one of the relevant lenses: database-enforced access control for multi-user and admin systems.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Performance Runtime

Use when a page must already inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Google Search Console as one of the relevant lenses: site verification, sitemap submission and search-performance monitoring.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: AI-Native Features

Reusable prompt packet for directly inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Structured Data / JSON-LD as one of the relevant lenses: machine-readable meaning for rich search and entity clarity.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Component System

Use it as a direct prompt inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use Core Web Vitals as one of the relevant lenses: LCP, INP and CLS as runtime quality signals.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Motion System

Use these pages directly inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use OWASP discipline as one of the relevant lenses: threat-aware build logic for secrets, authorization and data exposure.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — Prompt Packet: Migration and Refactor

Paste-ready prompt packet for working inside Lovable as reusable build doctrine.

PURPOSE. This prompt packet is designed to make Lovable produce a contemporary, production-aware result instead of a generic prototype. It must be pasted into the existing project, preferably after a Plan mode diagnosis, and adapted with the concrete brand, route and database names. The engine should use SMTP / Branded Email as one of the relevant lenses: relationship infrastructure with SPF, DKIM, DMARC and consent.

DELIVERABLES. Add or update components, data tables, routes, metadata, structured data, forms, admin screens, tests and success/error states. Include implementation notes in the project knowledge file. If the feature touches sensitive data, verify RLS and server-side auth. If it touches search, run the SEO/AEO review and fix metadata, sitemap, robots, llms.txt, Open Graph, canonical URLs, headings and schema. If it touches email, include SPF/DKIM/DMARC guidance, consent, unsubscribe and suppression logic.

ENGINE CONTROL NOTES. When Lovable proposes a fast but vague result, push it back into specificity: ask for exact route names, component boundaries, database tables, RLS policies, states, event logging, page metadata and verification commands. When the result is visually good but operationally weak, ask it to wire the visible action to a durable record or admin workflow. When the result is technically plausible but aesthetically generic, ask for editorial hierarchy, light-field composition, intelligent whitespace, branded motion tokens and non-template section geometry.

PROMPT. You are editing an existing Lovable project. Do not create a separate project. First inspect the current structure: routes, components, data tables, auth, admin routes, SEO metadata, public pages, forms, and integration points. Then propose a safe plan. Implement in focused batches. Preserve the current visual identity unless the requested section explicitly changes it. Build a corporate editorial cinematic interface with bright premium palette, semantic HTML, accessible components, reusable Tailwind tokens, Motion-based micro-interactions where useful, and strict no-generic copy. Every visible block must connect to a hidden operational purpose: lead capture, CRM memory, email sequence, analytics event, security boundary, or knowledge graph.

VERIFICATION. Request browser testing for the public flow, frontend tests for stateful components, edge-function verification for backend logic, and a security review before publishing. Then produce a compact change log: what changed, where, what to test, what remains manual.

Appendix — MAWIM Scorecard

Use these pages directly inside Lovable as reusable build doctrine.

MAWIM SCORECARD. Score every web app from 0 to 100. Meaning Layer 12: the site has a clear doctrine and specific promise. Visual Layer 12: the design is bright, corporate, cinematic and disciplined. Interaction Layer 10: motion and states guide understanding. Semantic Layer 12: SEO/AEO/GEO foundations are implemented. Conversion Layer 12: forms and CTAs capture qualified intent. Data Layer 10: records, events and consent are stored. Operational Layer 10: admin and CRM make work visible. Security Layer 12: auth, RLS, secrets and policies are reviewed. Evolution Layer 10: analytics, testing and iteration loops exist. Under 70, the app is a prototype. 70-84, it is usable but not sovereign. 85-94, it is professional. 95+, it is a living web intelligence system.

IMPROVEMENT LOOP. For each layer under 8/12 or 7/10, create one repair prompt. Do not ask for generic improvement. Ask for a named intervention: stronger H1, better lead form states, RLS policy review, campaign suppression list, mobile typography repair, hero hierarchy rewrite, page-specific JSON-LD, component extraction, or edge-function verification. The scorecard is a steering wheel, not a decoration.

SCORING PROCEDURE. Do not score from taste alone. Open the live page, inspect the route map, submit a test lead, read the database record, check admin visibility, run the SEO/AEO review, inspect mobile layout, verify focus states and ask Lovable for browser testing on the primary flow. Only then assign numbers. A beautiful static page with no durable memory cannot pass the Data Layer. A strong admin panel with exposed rows cannot pass the Security Layer. A rich blog without schema, internal links and llms.txt cannot pass the Semantic Layer.

ANTI-GENERIC COURT. Put every page on trial. Charge one: the headline could belong to any competitor. Charge two: the section has no data destiny. Charge three: the visual system uses fashionable effects without meaning. Charge four: the CTA is vague. Charge five: the admin has no operator value. Charge six: the blog is a list rather than a knowledge graph. Charge seven: the form collects contact data but not business context. Charge eight: the project has no test plan. Charge nine: the security model is assumed, not verified. Charge ten: the result cannot be understood by a crawler, a user or an operator without guessing. The verdict must be redesign, repair or removal.

SENTENCE. If the block fails only visually, repair composition. If it fails operationally, wire it to CRM, event tracking or admin. If it fails semantically, rewrite headings, metadata, FAQ and schema. If it fails security, stop publication until auth, RLS and secrets are reviewed. If it fails all four, delete the block and rebuild from first principles.

CROSS-EXAMINATION. Ask the page to defend itself. What is its unique claim? What does the visitor know after reading it that they did not know before? What does the operator receive after the visitor acts? What data becomes available? What next action becomes easier? What search entity does this page strengthen? What risk did the section introduce? What did motion clarify? What did it distract from?

IMPLEMENTATION ROADMAP. Week one: build the public core, brand system, homepage, services, lead magnet and SEO base. Week two: build knowledge hub, article templates, schema, llms.txt and internal links. Week three: build Supabase data spine, auth, admin leads, CRM pipeline and analytics events. Week four: add email templates, SMTP/branded email configuration, consent, unsubscribe, suppression and campaign views. Week five: run Lovable SEO/AEO review, security review, RLS review, browser testing and performance checks. Week six: publish to custom domain, connect Google Search Console, submit sitemap, monitor leads, refine copy and create the next content cluster.

MEASUREMENT. At the end of each week, record what changed in four columns: visible quality, semantic discoverability, operational memory and risk reduction. A week that improves beauty but adds no memory is incomplete. A week that adds backend but weakens user clarity is incomplete. The mature web app advances on all axes: beauty, meaning, data, security and learning.

LOVABLE CADENCE. Do not push the engine in one massive unstable request after the first build. Work like a systems architect: inspect, plan, implement one layer, verify, commit or keep a stable version, then continue. Use project knowledge for durable rules: brand palette, component naming, no generic copy, admin security, SEO standards and testing habits. Use skills or reusable prompts for recurring tasks: article creation, service page expansion, CRM screen creation, and RLS review.

FINAL MANIFESTO. The future web app is not a set of screens. It is a visible-hidden intelligence organism. The visible layer earns attention through cinematic corporate clarity. The hidden layer earns trust through data, auth, security, events and verification. Lovable.dev and similar engines compress the distance between imagination and running software, but compression creates danger: generic output, weak data models, exposed admin, fake polish, untested flows. The Moussaif method answers with pressure: name the ontology, design the block, specify the data, command the engine, test the runtime, secure the memory, publish the semantic graph, and evolve through evidence. The builder becomes Picasso only when beauty obeys architecture and architecture obeys truth.

HIDDEN SCIENCE. The hidden science is not merely database plumbing. It is memory and permission. A form without a table is amnesia. A table without RLS is exposure. A dashboard without next actions is vanity. An email system without consent and suppression is reputation damage. A blog without schema and internal links is isolated text. The hidden layer decides whether the visible promise is true.

FINAL RULE. Build nothing that cannot be read by humans, crawled by search engines, understood by AI answer systems, operated by admins, secured by policies, measured by events and improved by evidence. This is the closing doctrine of the codex: the web app is alive only when its appearance, intelligence and runtime form one accountable system.

VISIBLE SCIENCE. The visible science is not decoration. It is the disciplined arrangement of perception: headline pressure, typographic authority, white space, motion, semantic sections, contrast, editorial rhythm and CTA gravity. A corporate cinematic page does not scream. It guides. It makes the visitor feel that the company has already organized the problem before the visitor speaks.

ENGINE SCIENCE. The engine understands what is made explicit. A weak prompt asks for beauty. A strong prompt specifies routes, users, components, states, data, policies, tests, metadata, and failure conditions. Lovable can move fast, but the builder must govern direction. The machine accelerates intent; it does not replace architectural judgment.