Below is a single, clinical source-of-truth you can paste into a new GPT Business account or share with the team. It captures what’s locked, current status, how IntegAI will take over every website API call, and pitfalls with practical mitigations. I’ve refined and tightened language where helpful, while keeping a relentless bias toward getting it done.

---

HISL Website Build — Locked-In Technical Blueprint

Version: v1.5 • Date: 2025‑09‑19

Owner: Website Build (HISL) • Primary Backend: IntegAI (offline‑first orchestration)

0) Status Snapshot (today)

Brand/Identity: Locked. HISL v1 mark (gold hard hat + embedded brain, green eco‑signal nodes, orbital paths, biocircuit overlays). Deployment kit + launch copy: ready.

Website Wireframe: Lock‑in is the gate. Directive stands: populate & lock each section across Whimsical,

Notion, NAS before the next build phase. A few sections remain to finalize; see “Open Items.”

IntegAI Platform: Phase P1 delivered, transitioning to P2 (LangGraph v0). Core locked items below.

Cutover Strategy: IntegAI to become the single API entry point for the site via proxy → shadow → phased takeover.

---

1) Hard-Locked Decisions (do not revisit)

Performance & Observability Targets (P1):

LLM TTFT ≤ 800 ms, P50 gen ≥ 40 tok/s (fast‑path Gemma via llama.cpp).

Perf harness & test suite: ready.

Agent Graph (LangGraph) — 9 Core Roles: Planner / Retriever / Judge / Executor / Memory / Safety / Tools / Observer / Publisher.

(Note: “Safety” is minimal for MVP; keep pass‑through unless prompted.)

Offline RAG Path: Obsidian → E5‑small‑v2 (GGUF) → Chroma + Neo4j (content graph + vector store).

LLM Gateway: llama.cpp Gemma fast‑path; Mistral‑7B Q4\_0 fallback.

Front Door: NGINX reverse proxy. (Auth stack—Keycloak, Vault—retained but de‑emphasized for MVP.)

Artifacting: .integpkg bundles to MinIO for model & data artifacts (promotion across envs).

Hardware: Lenovo host active; RAVEN GPU inbound (enable higher‑throughput path when online).

---

2) What “IntegAI takes over all website API calls” means

End state: The website never calls third‑party services directly. It calls

only. IntegAI then orchestrates content retrieval, generation, search, forms, and publishing—optionally delegating to upstreams via adapters.

2.1 API Ownership Model

Single entry: https://site.hisl.ai/api/\* → NGINX → IntegAI Gateway.

Adapters (thin): Content, Assets, Search, Forms, CMS, Email/CRM.

Adapters allow gradual retirement of legacy calls while keeping one surface for the website.

Policy & Routing: LangGraph Planner + Router decides:

local RAG vs. cached content vs. legacy adapter vs. LLM generation.

2.2 Minimal HTTP Surface (stable contracts)

> Keep these contracts stable so the website never refactors again.

Read (GET):

/v1/site/page?slug={slug} → {title, hero, sections[], metadata, lastUpdated}

/v1/site/menu → {primary[], footer[]}

/v1/site/search?q={q}&limit=&after= → {results[{type, title, url, excerpt, score}], next}

/v1/site/assets/{id} → 302 redirect to canonical (MinIO/CDN) or {signedUrl} (fallback)

/v1/site/seo?slug={slug} → {title, description, keywords[], og:{...}, schemaOrg:{...}}

Write (POST):

/v1/site/forms/{formId} → {ok, idempotencyKey, triage:{tag,severity}, route:{owner,channel}}

/v1/site/generate/copy (template‑driven) → {draft, tokens, cost, provenance}

/v1/site/publish (Publisher agent) → {ok, artifact:.integpkg, urls[]}

Infra (GET) (optional but handy):

/v1/site/health → {uptime, cache, gpu, qps, p95}

/v1/site/trace/{id} → link to Langfuse/Phoenix trace (for dev only)

Contract rules (MVP):

Timeouts: 300ms for cached reads; 2.5s for non‑cached; 6–10s for generation endpoints.

Idempotency: All POSTs accept Idempotency-Key.

Versioning: Default v1; breaking changes only under /v2.

---

3) Execution Plan (phased takeover)

Phase 0 — Mirror (today)

Add NGINX map: /api/\* → IntegAI (shadow mode).

Request teeing: For READs, IntegAI builds responses but does not serve them yet; it logs parity against legacy responses.

Cache warmers: Crawl wireframed slugs → prebuild /v1/site/page cache.

Phase 1 — Proxy + Cache (GET first)

Flip GET routes to IntegAI (pages, menu, search, seo).

Fallback chain: Cache → Local RAG → Legacy adapter.

Traffic weighting: 100% to IntegAI for GET; POSTs still legacy.

Phase 2 — Mutations (POST)

Move forms and generate/publish to IntegAI.

Ensure idempotency & retries at gateway.

Enable Publisher agent to write artifacts (.integpkg) to MinIO, then invalidate caches/CDN.

Phase 3 — Decommission

Remove direct legacy integrations in website.

Keep adapters in IntegAI for historical backfills only.

Go/No‑Go checks (each phase):

p95 latency within budget, error rate < 0.5%, parity > 99% for deterministic reads, 0 duplicate submissions for POSTs (idempotency).

---

4) Website Build Scope (what’s locked vs. open)

4.1 Locked/In-Place

Brand: HISL v1 mark + deployment kit.

Source of Truth: Wireframe sections live in Whimsical; copy blocks in Notion; canonical assets in NAS.

IntegAI Readiness: Gateway, perf harness, agent roles, offline RAG path, artifacting path, NGINX front door—locked.

4.2 Open Items (blocking next phase)

Wireframe Section Lock‑In: Every page section must be finalized (IDs, data fields, and component mapping).

Menu & Routing Table: Finalize slugs and canonical URLs (used by cache warmers & SEO endpoint).

Form IDs + Schemas: Lock formId and required fields per form (contact, RFP, newsletter, careers).

Asset Canonicalization: Ensure every on‑page image/video maps to a single assetId in NAS/MinIO.

---

5) IntegAI Internals (how it works for the website)

5.1 Data Flow (read path)

1. Website → /v1/site/page?slug

2. NGINX → IntegAI Gateway

3. Gateway checks L1 memory cache → L2 Redis (optional)

4. Planner chooses:

Cache hit, return.

Local RAG (Obsidian → E5‑small‑v2 GGUF → Chroma + Neo4j) to assemble section content.

Legacy adapter (until decommission).

5. Judge validates draft (basic constraints: token length, required fields present).

6. Observer logs perf minimal to Phoenix / Langfuse (dev‑only).

5.2 Data Flow (write path)

1. Website POST → forms/{formId} with Idempotency-Key.

2. Executor writes normalized record → Postgres (or MinIO JSON snapshot).

3. Memory tags + Tools route to CRM/email as configured (thin adapter).

4. Response returns triage + owner route. (No analytics/tracking ceremony in MVP.)

5.3 Generation & Publishing

/generate/copy is template‑driven (component slots + tone).

Publisher emits .integpkg with:

/content/\*.mdx or JSON blocks

/assets/\* references

/manifest.json (slugs, deps, checksum)

Deploy hook: On new artifact → invalidate caches & (optionally) trigger site rebuild if SSR/SSG is in use.

---

6) Minimal Contracts the Website Should Assume (front-end agnostic)

All content is block‑structured, keyed by sectionId with component + props.

SEO is API‑driven via /v1/site/seo?slug to centralize titles/OG/schema (one source of truth).

Search is semantic-first; results return type (page, doc, asset) with canonical url.

Forms are idempotent and return triage hints; front end only displays confirmation.

---

7) Performance Budgets (keep us honest)

Static/cached reads: TTFB ≤ 200 ms (edge+L1).

RAG‑backed reads: ≤ 2.0 s p95.

LLM generation endpoints: ≤ 10 s p95 (until GPU online).

Throughput (post‑GPU): target ≥ 15 RPS sustained on gen endpoints with batch prompts.

---

8) Foldering & Content Ops (NAS + Artifacts)

/HISL

/Brand # logo pack, color, type, do/don'ts

/Wireframes # Whimsical exports per page/section

/Copy # Notion exports -> MD/JSON canonical

/Assets # images/video; assetId = filename stem

/IntegAI

/Artifacts # .integpkg releases (immutable)

/Promote # staging -> prod via checksum list

Rule: Website only consumes /IntegAI/Artifacts (immutable). Anything mutable is upstream of Publisher.

---

9) Acceptance Criteria (to close each gate)

Wireframe Lock: 100% sections have sectionId, component mapping, and content fields enumerated.

Menu/Routes: All slugs canonical with redirects defined; no TBD.

Forms: Schemas frozen; idempotency keys validated under load (no dupes @ 5 RPS).

API Parity: GET endpoints parity ≥ 99% byte‑for‑byte vs. legacy for deterministic pages.

Cutover: Error rate < 0.5% during 24h soak; cache hit ≥ 70% for high‑traffic slugs.

---

10) Pitfalls & Practical Mitigations (no process theater)

Wireframe drift (design updates after lock):

→ Mitigate with component registry (component, required props); rejects unknown props at publish time.

Slug instability (SEO loss):

→ Freeze slug in manifest; force redirects map in /v1/site/seo.

Cold starts (first hit is slow):

→ Run cache warmers nightly across top routes; pre‑hydrate RAG context windows for hero pages.

Generation variability (LLM drift):

→ Template + deterministic decoding on critical sections; keep “creative” only for blog intros.

GPU dependency risk (RAVEN arrival timing):

→ Keep llama.cpp CPU path as safe baseline; batch prompts for heavy pages; pre‑generate long‑form content.

Adapter debt (never decommissioned):

→ Enforce deprecation dates per adapter; CI warns if called after end‑of‑life.

---

11) ENV & Minimal Config (so it boots)

INTEGAI\_MODE=prod|staging|shadow

INTEGAI\_CACHE\_TTL\_PAGES=300

INTEGAI\_GENERATION\_TIMEOUT\_MS=10000

INTEGAI\_MINIO\_BUCKET=integpkgs

INTEGAI\_OBSIDIAN\_PATH=/data/obsidian

INTEGAI\_VECTOR\_INDEX=chroma://...

INTEGAI\_GRAPH\_DB=neo4j://...

INTEGAI\_LLM\_FASTPATH=gemma-... (llama.cpp)

INTEGAI\_LLM\_FALLBACK=mistral-7b-q4\_0

INTEGAI\_FORMS\_DB=postgres://... (or MinIO JSON if DB deferred)

INTEGAI\_ALLOWED\_ORIGINS=https://www.hisl.\*

NGINX\_UPSTREAM=http://integai:8080

(Security variables like secrets/tokens exist but remain out of scope for MVP docs.)

---

12) Simple Routing Map (NGINX → IntegAI)

/api/v1/site/\* → IntegAI (shadow → live)

/assets/\* → MinIO/CDN (or 302 via /v1/site/assets/{id})

/healthz → IntegAI health (internal)

---

13) Test Matrix (kept minimal)

Parity tests (GET): For each slug, diff legacy vs IntegAI JSON; fail if >1% delta.

Load smoke: k6 @ 100 VU for 5 min on top 20 pages; p95 under budget.

Form idempotency: 100 concurrent POSTs with same Idempotency-Key → exactly 1 stored.

Failure drills: Kill RAG vector store → Gateway must serve cache or legacy adapter within SLA.

---

14) “DECISION / SHIP / BLOCKER” (current)

SHIP: Perf harness; Gateway; 9‑core agent graph baseline; Offline RAG; .integpkg artifacting; NGINX front door.

DECISION: Finalize forms schemas + menu/routes; confirm cache TTLs (suggest 5–10 min).

BLOCKER: Wireframe section lock‑in (IDs + props). RAVEN GPU not yet online (nice‑to‑have, not gating).

---

15) Minimal Front‑End Contract (to hand to web team)

Always fetch pages from /v1/site/page?slug. Render by component using mapped props.

Fetch SEO via GET /v1/site/seo?slug and set tags accordingly.

Forms POST to /v1/site/forms/{formId} with an Idempotency-Key header.

Do not call legacy services directly. All traffic goes to /api/v1/site/\*.

---

16) Paste‑Ready “System Prompt” for your GPT Business Account

> Role: HISL Website API Orchestrator (IntegAI)

Mission: Be the single source of truth for the HISL website build. Generate, validate, and publish website content via stable API contracts. Prioritize delivery, performance, and correctness. Minimize security/tracking ceremony unless explicitly requested.

Locked Context (must honor):

Performance targets: TTFT ≤ 800 ms, P50 ≥ 40 tok/s (Gemma fast‑path; Mistral‑7B Q4\_0 fallback).

Agent graph roles: Planner/Retriever/Judge/Executor/Memory/Safety/Tools/Observer/Publisher.

Offline RAG: Obsidian → E5‑small‑v2 GGUF → Chroma + Neo4j.

Gateway contracts (v1):

GET /v1/site/page?slug → block‑structured page.

GET /v1/site/menu, /v1/site/search, /v1/site/seo?slug, /v1/site/assets/{id}.

POST /v1/site/forms/{formId}, /v1/site/generate/copy, /v1/site/publish.

Artifacting: output .integpkg → MinIO; immutable; triggers cache invalidation.

NGINX front door; all website calls go through IntegAI.

Behavioral Rules:

Respond only with data structures defined in v1 contracts, or with precise operational instructions.

Prefer cache → local RAG → adapter in that order.

Enforce idempotency on POST; reject unknown sectionId or component props.

Keep SEO canonical in /v1/site/seo; disallow title/OG drift in page payloads.

When uncertain, return the most recent published artifact rather than regenerating.

Non‑Goals (MVP): advanced security flows, behavioral analytics, A/B infra.

Success: website runs solely on IntegAI APIs; parity met; pages fast; forms rock‑solid.

---

17) Quick “Do‑Next” List (to trigger cutover)

1. Freeze wireframe sections (IDs + props) and export to NAS.

2. Lock slugs + redirects + menu; point cache warmer at the list.

3. Freeze forms schemas + issue public formIds.

4. Point website /api/\* at IntegAI (shadow → GET live) and run parity soak.

5. Flip POST routes; verify idempotency; decommission legacy calls.

Generic Agents IntegAI will build /feature for deployment on the website, these will have integAI’s core dna (ethical layer)

IntegAI Sector → Agents Matrix

Construction

Agents: EHS Agent, Project PM Agent, BIM/Design Agent

Functions: Site safety & compliance, risk reporting, scheduling & resource planning, blueprint analysis & cost estimation.

Environmental

Agents: ESG Agent, Sustainability Metrics Agent

Functions: Emissions tracking, biodiversity monitoring, ESG reporting, lifecycle analysis.

Conservation / SPAB

Agents: Heritage Preservation Agent, Planning Approvals Agent

Functions: Restoration planning, heritage asset scanning, permit workflows, SPAB guidance.

Pharma

Agents: QA Agent, Clinical R&D Orchestrator (CROx)

Functions: GMP compliance, audit preparation, document QC, protocol co-pilot, recruitment simulation, monitoring hub.

Public Procurement

Agents: Procurement Agent, Anti-Corruption Oversight Agent

Functions: Tender evaluation, bid drafting, compliance guardrails, transparency enforcement, anomaly detection, audit reporting.

Agriculture

Agents: Crop Monitoring Agent, Agri-Supply Agent

Functions: Soil/irrigation monitoring, yield optimization, sustainability metrics, logistics planning, pricing intelligence.

Trader / Economics

Agents: Market Signals Agent, Risk & Portfolio Agent

Functions: Market signals aggregation (equities, FX, commodities), portfolio modeling, risk detection & hedging strategies.

Healthcare / Pet Care

Agents: Clinical Workflow Agent, Pet Care Agent

Functions: Clinical/pet health notes summarization, triage workflows, compliance with health records, diagnostics & treatment planning.

Insurance

Agents: Ops Agent, Fraud Detection Agent

Functions: Claims processing, policy administration, fraud detection & anomaly flagging.

Custom

Agents: Wizard Agent, Child Agents

Functions: Bespoke orchestration, sector-specific customization, callback learning (DP + HE).