0) TL;DR (locked today)

Perf targets (P1): TTFT ≤ 800 ms (Gemma fast‑path via llama.cpp), P50 ≥ 40 tok/s. Stack: llama.cpp (CPU), Gemma fast‑path; Mistral‑7B Q4\_0 fallback; LangGraph v0 (P2); Phoenix, Langfuse; MinIO, Chroma, Neo4j, Postgres; Keycloak, Vault, NGINX. Host: Lenovo; RAVEN GPU inbound.

9‑core agent graph: Planner / Retriever / Judge / Executor / Memory / Safety / Tools / Observer / Publisher.

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

Security front‑door: NGINX + Keycloak, secrets via Vault.

Artifacting: .integpkg bundles to MinIO.

Single Source of Truth: IntegAI LIVE BUILD repo + Obsidian vault.

> DECISION: Above are locked. Everything else below aligns to these and is framed with DECISION / SHIP / BLOCKER where relevant.

---

1) Operating Principles (for GPT System Prompt)

Purpose: Sovereign, offline‑first orchestration with safe inference, observability, and reproducible artifacts.

Behavioral Contract (embed verbatim in your GPT “System Prompt”):

1. Offline‑first default. Prefer local indexes (Chroma/Neo4j) and internal artifacts. Never fetch the public web unless explicitly permitted via the tools.web\_fetch and policy gate safety.allow\_online=true.

2. Deterministic pipelines. Use the 9‑core graph. Planner proposes → Retriever gathers → Judge checks → Executor acts → Memory writes → Safety gates → Tools perform → Observer logs → Publisher emits artifacts.

3. Latency discipline. Keep TTFT ≤ 800 ms by default. Use Gemma via llama.cpp for first draft; escalate to Mistral‑7B Q4\_0 only when required by complexity or length (Judge decides).

4. Content safety & compliance. Apply safety.policy\_profile rules. Decline chain‑of‑thought; provide final answers + brief reasoning. No background tasks.

5. Telemetry minimalism. Log only non‑sensitive spans/metrics via observer.log\_event. Use redaction for PII.

6. DECISION/SHIP/BLOCKER tags. When making recommendations or status calls, clearly tag items.

7. Artifacts first‑class. Package outputs as .integpkg (manifest + payload) to MinIO; publish pointers, not blobs, in chat.

---

2) Agent Graph (LangGraph‑style) — roles & contracts

State keys (global):

goal, query, context, constraints, policy, plan, docs, graph\_hits, draft, verdict, actions, artifacts, telemetry, memory\_events, publish\_spec.

Nodes (locked):

1. Planner

Input: goal, query, constraints, policy.

Output: plan (steps, required tools, data sources, latency budget).

DECISION: must propose offline path first; include cost/latency estimate.

2. Retriever

Input: plan.

Actions: vector\_search.query, graph.query, minio.get (as needed).

Output: docs (ranked contexts), graph\_hits.

SHIP: hybrid retrieval (dense + graph neighborhood + rule filters).

3. Judge

Input: plan, docs, graph\_hits.

Output: verdict (sufficiency, trust score, escalate\_to\_mistral?, requires\_HITL?).

Rules: If trust < τ (default 0.72) → request more context or HITL.

4. Executor

Input: plan, verdict.

Actions: tools.\* according to plan (e.g., transform docs, generate draft).

Output: draft, actions taken.

5. Memory

Input: draft, docs, actions.

Actions: memory.write (summary, embeddings, graph edges).

Output: memory\_events.

6. Safety

Input: draft, actions, policy.

Output: draft\_safe, redactions, allow\_online (bool).

DECISION: disallow online unless policy + user grant.

7. Tools (broker)

Provides access to function endpoints (MinIO/Chroma/Neo4j/Vault/etc.).

Enforces timeouts/retries; records to Observer.

8. Observer

Input: all spans.

Action: observer.log\_event.

Output: telemetry (span IDs, metrics).

9. Publisher

Input: draft\_safe, artifacts.

Actions: package as .integpkg, minio.put, emit publish\_spec.

Output: final user answer + artifact links.

---

3) Tool (Action) Schemas for GPT Business

> Paste these as Actions (function calling) in your GPT Business. They’re minimal, composable, and offline‑first. Namespaces imply routing in your gateway.

3.1 vector\_search.query

{

"name": "vector\_search.query",

"description": "Offline dense retrieval from Chroma. Use for Obsidian-ingested content. Returns ranked chunks.",

"parameters": {

"type": "object",

"properties": {

"collection": {"type": "string", "description": "Chroma collection name", "enum": ["obsidian\_main", "assets\_code", "sops"]},

"query": {"type": "string"},

"top\_k": {"type": "integer", "default": 8, "minimum": 1, "maximum": 50},

"filters": {"type": "object", "description": "Metadata filters (exact match only)"},

"include\_payload": {"type": "boolean", "default": true}

},

"required": ["collection", "query"]

}

}

3.2 graph.query

{

"name": "graph.query",

"description": "Offline graph lookup in Neo4j. Safe subset of Cypher.",

"parameters": {

"type": "object",

"properties": {

"cypher": {"type": "string", "description": "Pre-approved Cypher template id or parameterized query"},

"params": {"type": "object", "description": "Named parameters"},

"limit": {"type": "integer", "default": 50, "minimum": 1, "maximum": 500}

},

"required": ["cypher"]

}

}

3.3 minio.get

{

"name": "minio.get",

"description": "Read object from MinIO (offline artifact store).",

"parameters": {

"type": "object",

"properties": {

"bucket": {"type": "string"},

"key": {"type": "string"},

"as\_text": {"type": "boolean", "default": false}

},

"required": ["bucket", "key"]

}

}

3.4 minio.put

{

"name": "minio.put",

"description": "Write object to MinIO. Use for .integpkg and derived artifacts.",

"parameters": {

"type": "object",

"properties": {

"bucket": {"type": "string"},

"key": {"type": "string"},

"content\_b64": {"type": "string", "description": "Base64 content"},

"content\_type": {"type": "string", "default": "application/octet-stream"},

"metadata": {"type": "object"}

},

"required": ["bucket", "key", "content\_b64"]

}

}

3.5 observer.log\_event

{

"name": "observer.log\_event",

"description": "Emit structured telemetry to Langfuse/Phoenix via gateway.",

"parameters": {

"type": "object",

"properties": {

"event": {"type": "string", "enum": ["span\_start","span\_end","metric","warning","error"]},

"name": {"type": "string"},

"ts": {"type": "string", "description": "ISO-8601"},

"data": {"type": "object"},

"pii\_redacted": {"type": "boolean", "default": true}

},

"required": ["event", "name", "ts"]

}

}

3.6 memory.write

{

"name": "memory.write",

"description": "Persist short summaries, keys, embeddings, and edges (Postgres + Neo4j).",

"parameters": {

"type": "object",

"properties": {

"kind": {"type": "string", "enum": ["summary","edge","tag","cache"]},

"payload": {"type": "object"}

},

"required": ["kind", "payload"]

}

}

3.7 security.get\_secret

{

"name": "security.get\_secret",

"description": "Fetch secret from Vault via policy-guarded path. Do not log values.",

"parameters": {

"type": "object",

"properties": {

"path": {"type": "string"},

"key": {"type": "string"}

},

"required": ["path","key"]

}

}

3.8 safety.check

{

"name": "safety.check",

"description": "Validate draft output against safety policy\_profile. Returns allow, redactions, notes.",

"parameters": {

"type": "object",

"properties": {

"policy\_profile": {"type": "string", "enum": ["default","strict\_enterprise","kidsafe"]},

"text": {"type": "string"}

},

"required": ["policy\_profile","text"]

}

}

3.9 publisher.package

{

"name": "publisher.package",

"description": "Create a .integpkg bundle and return a MinIO URL pointer.",

"parameters": {

"type": "object",

"properties": {

"title": {"type": "string"},

"manifest": {"type": "object"},

"files": {

"type": "array",

"items": {"type": "object", "properties": {

"path": {"type": "string"},

"content\_b64": {"type": "string"},

"content\_type": {"type": "string"}

}, "required": ["path","content\_b64"]}

},

"bucket": {"type": "string"},

"key": {"type": "string"}

},

"required": ["title","manifest","files","bucket","key"]

}

}

3.10 tools.transform

{

"name": "tools.transform",

"description": "Local transforms: chunk, summarize, extract JSON, reformat, diff.",

"parameters": {

"type": "object",

"properties": {

"op": {"type": "string", "enum": ["chunk","summarize","extract\_json","reformat","diff"]},

"input": {"type": "string"},

"spec": {"type": "object"}

},

"required": ["op","input"]

}

}

> OPTIONAL (kept disabled by default in sovereign mode)

web\_fetch.get (online access; requires safety.allow\_online=true) and executor.shell (sandboxed scripts with a safe allowlist). Include only if you deliberately enable.

---

4) Data & Indexing Schema (offline path)

4.1 Obsidian → Chroma (E5‑small‑v2 GGUF)

Chunking: 512–768 tokens; overlap 64.

Metadata: source\_path, doc\_id, title, created\_ts, updated\_ts, tags[], pii=false|true, version.

Collections:

obsidian\_main (notes, specs, SOPS) — DECISION

assets\_code (snippets, manifests) — DECISION

sops (procedures; high signal) — SHIP

4.2 Neo4j Knowledge Graph

Node labels: Doc, Concept, Entity, Task, Artifact.

Core relationships:

(Doc)-[:MENTIONS]->(Entity)

(Concept)-[:RELATES\_TO]->(Concept) (weighted)

(Task)-[:USES]->(Artifact)

(Doc)-[:DERIVED\_FROM]->(Artifact)

Minimal node props: id, title|name, kind, importance (0..1), updated\_ts.

Index: BTREE on id; text index on title|name.

Guardrails: Only allow pre‑approved Cypher templates via graph.query.cypher IDs.

4.3 Postgres (Memory + Eval)

Tables (DECISION):

memory\_summary(id, key, value\_json, created\_ts)

memory\_edge(src\_id, rel, dst\_id, weight, created\_ts)

eval\_runs(id, suite, model, metrics\_json, created\_ts)

prompt\_cache(key\_hash, value\_json, ttl\_ts)

---

5) Performance & Reliability Profile

Fast‑path: Gemma via llama.cpp (Q? with TTFT ≤ 800 ms).

Fallback: Mistral‑7B Q4\_0 (Judge triggers on complexity/length).

Retries/Timeouts (Tools broker):

timeout: 800 ms local calls, 2 s MinIO/Chroma, 3 s Neo4j.

retries: 1 (idempotent ops), exponential backoff (100 ms).

Determinism knobs: temperature 0.2 (fast‑path), 0.4 (fallback); max\_tokens budgets per node.

Prompt caching: enable prompt\_cache (Postgres) keyed by normalized plan.

---

6) Security Front‑Door

Ingress: NGINX → Keycloak (OIDC) → LLM Gateway.

AuthZ: per‑namespace scopes: vector:read, graph:read, minio:read/write, observer:write, memory:write, security:read.

Secrets: Vault KV v2; never log secret values; return handles.

PII: observer.log\_event.pii\_redacted=true default.

Air‑gap control: safety.policy\_profile=strict\_enterprise disallows web\_fetch.get.

---

7) Observability (Phoenix + Langfuse)

Spans: planner, retriever.vector, retriever.graph, judge, executor.gen, safety, publisher, tool.<name>.

Metrics: ttft\_ms, tok\_per\_s, ctx\_tokens, retrieval\_precision@k, verdict\_trust, errors.

Sampling: 100% for P1, adjust later.

Redaction: regex for emails, phones, IDs; hashing for doc\_ids.

---

8) .integpkg Artifact Spec (DECISION)

Structure (zip):

/manifest.json

/payload/<files...>

manifest.json fields:

title, version, created\_ts, model, plan\_hash, inputs\_summary, outputs\_summary, data\_lineage[], checksums, policy\_profile, metrics.

Storage: MinIO bucket integ-artifacts/YY/MM/<uuid>.integpkg.

---

9) Configuration Manifests (one‑pager YAML)

Use this as a source‑of‑truth config; your gateway reads it and maps to tooling.

integai:

profile: "sovereign"

models:

fast\_path:

name: "gemma"

runner: "llama.cpp"

temp: 0.2

max\_tokens: 768

fallback:

name: "mistral-7b-q4\_0"

runner: "llama.cpp"

temp: 0.4

max\_tokens: 2048

perf:

ttft\_ms\_target: 800

p50\_tok\_per\_s: 40

retrieval:

chunk\_tokens: 640

dense\_model: "e5-small-v2-gguf"

collections: ["obsidian\_main","assets\_code","sops"]

top\_k: 8

graph:

cypher\_templates:

concept\_neighbors: "MATCH (c:Concept {name:$name})-[:RELATES\_TO]->(n) RETURN n LIMIT $limit"

safety:

policy\_profile: "strict\_enterprise"

allow\_online: false

storage:

minio\_bucket: "integ-artifacts"

chroma\_host: "localhost:8001"

neo4j\_uri: "bolt://localhost:7687"

security:

keycloak\_realm: "integ"

vault\_prefix: "kv/integ"

telemetry:

langfuse\_project: "integai"

phoenix\_endpoint: "http://phoenix.local"

---

10) GPT Business “System Prompt” (drop‑in)

> Title: IntegAI – Sovereign Offline‑First Orchestrator (vP1)

You are IntegAI, a sovereign, offline-first AI orchestrator. Follow these rules:

1) OFFLINE-FIRST: Use only local tools and indexes unless safety policy explicitly allows online. If unsure, ask the Safety node via `safety.check`.

2) GRAPH OF NODES: Execute work through the 9-node plan (Planner → Retriever → Judge → Executor → Memory → Safety → Tools → Observer → Publisher). Keep a tight loop; do not perform background work.

3) LATENCY: Prefer the fast-path model. Keep TTFT ≤ 800 ms; keep responses concise unless the user asks for exhaustive detail.

4) SAFETY: No chain-of-thought. Provide final answers and brief, verifiable steps. Respect `policy\_profile`. Decline disallowed content.

5) ARTIFACTS: Package substantive outputs as `.integpkg` via `publisher.package`, then return links (not blobs).

6) TELEMETRY: Emit minimal, redacted spans via `observer.log\_event`.

7) TAGGING: When making operational calls or recommendations, use DECISION / SHIP / BLOCKER tags.

---

11) Example “happy path” flow (internal)

1. Planner: produce 3–5 step plan with offline sources.

2. Retriever: vector\_search.query(top\_k=8) + graph.query(concept\_neighbors) → merge rerank.

3. Judge: trust ≥ 0.72? If not, expand context or HITL flag.

4. Executor: generate draft using Gemma; mark “fallback\_needed=false”.

5. Safety: safety.check with strict\_enterprise.

6. Publisher: assemble .integpkg (manifest + derived outputs) → minio.put → return link.

---

12) Evaluation Harness (locked & refined)

Suites: Q&A factual, SOP generation, JSON extraction, code docstring, retrieval recall, long‑context summarization.

Metrics: EM/F1 for Q&A; Structural F1 for JSON; Rouge‑L for summarization; Retrieval P@k, nDCG; latency metrics; refusal correctness.

Acceptance gates (P1):

Retrieval P@8 ≥ 0.55, nDCG@8 ≥ 0.62 on Obsidian corpus.

JSON extraction structural F1 ≥ 0.92.

SOP generation passes checklist ≥ 90%.

---

13) Enhancements & Refinements (recommended)

DECISION (adopt now):

Prompt cache & response cache at gateway keyed by (normalized\_plan, user\_query, policy\_profile) → improves TTFT 10–25%.

Grammar-constrained JSON (llama.cpp --grammar) for extract\_json tasks → boosts structural F1, reduces post‑hoc fixes.

Hybrid rerank (dense score + graph prior) → stable trust scores and less drift.

SHIP (next 1–2 days):

KV cache reuse across retries and node hops in llama.cpp.

Quantization audit (Gemma Q4\_K\_M vs Q5) to balance TTFT vs recall for long answers.

Phoenix playbooks for drift and hallucination triage (auto‑open when verdict\_trust < 0.6).

MinIO lifecycle: auto‑expire raw intermediates in 7 days; keep .integpkg forever unless flagged.

BLOCKER (needs hardware / access):

GPU bring‑up (RAVEN): validate GGUF compatibility; test Flash‑Decoding style kernels; target P90 ≥ 65 tok/s.

Keycloak realms hardening (rotation, mTLS to gateway).

HITL console for Judge escalations (simple web pane).

---

14) Safety Profiles (ready to use)

default: standard enterprise disallows online unless user grants; mild redaction.

strict\_enterprise (DECISION default): no online; redaction aggressive; code execution disabled; tool outputs rate‑limited.

kidsafe: extra content filters; restricts some transformation ops.

---

15) Minimal SOPs (embed in Obsidian → sops collection)

SOP‑001: Ingestion — Obsidian → chunk (640/64) → embed (E5‑small‑v2 GGUF) → Chroma upsert (with metadata) → Graph extraction (NER + linking) → Neo4j upsert.

SOP‑002: Release a Package — create manifest → checksum files → publisher.package → minio.put → reply with link + DECISION/SHIP status.

SOP‑003: Incident Triage — if errors or trust < 0.6: capture spans → Phoenix report → annotate BLOCKER → open HITL ticket.

---

16) Concrete Example Prompts (to seed GPT Business “Instructions Library”)

“IntegAI Status Pulse”

“Summarize current DECISION/SHIP/BLOCKER, then list top 3 risks and mitigations. Do not fetch online. Return a .integpkg manifest + final summary.”

“SOP Builder”

“From Obsidian sources tagged sop, build a single SOP with checklist and owner. Output JSON schema + Markdown, then package.”

“RAG Q&A (Sovereign)”

“Answer strictly from local indexes; cite source\_path and doc\_id for each claim; if insufficient, say so.”

---

17) Governance & Data Retention

PII default off. If PII detected, redact and store only pointers.

Retention: Intermediates (7 days), Artifacts (indefinite), Telemetry (30 days; metrics only).

Access: Principle of least privilege via Keycloak scopes per tool.

---

18) Risks & Mitigations

Risk: Retrieval drift with vault growth → Mitigation: nightly eval + topic routers in graph.

Risk: Latency spikes on fallback → Mitigation: Judge pre‑scores complexity; pre‑warm fallback runner.

Risk: Policy bypass by tool misuse → Mitigation: Safety node gates allow\_online; broker enforces scope.

---

19) Quick “Inject into GPT Business” Checklist

Paste System Prompt (Section 10).

Create Actions for tools in Section 3 (enable only those you need now).

Set tool secrets via your infra (the GPT itself should call security.get\_secret, not hold them).

Provide starter Knowledge (upload .integpkg manifests or Obsidian exports if using GPT file retrieval as a stopgap).

Test with the three Example Prompts (Section 16) and verify spans in Langfuse/Phoenix.

---

20) Reference Cypher Templates (safe list)

concept\_neighbors(name, limit)

doc\_by\_tag(tag, limit) → MATCH (d:Doc)-[:HAS\_TAG]->(:Tag {name:$tag}) RETURN d LIMIT $limit

artifact\_lineage(id) → MATCH (a:Artifact {id:$id})-[:DERIVED\_FROM\*0..2]->(x) RETURN a,x

---

21) JSON Schemas (common)

Publish Spec

{

"type": "object",

"properties": {

"title": {"type":"string"},

"summary": {"type":"string"},

"artifacts": {

"type":"array",

"items": {"type":"object", "properties": {

"bucket":{"type":"string"},

"key":{"type":"string"},

"content\_type":{"type":"string"},

"sha256":{"type":"string"}

}, "required":["bucket","key"]}

}

},

"required": ["title","artifacts"]

}

Judge Verdict

{

"type":"object",

"properties":{

"sufficient":{"type":"boolean"},

"trust":{"type":"number"},

"fallback\_needed":{"type":"boolean"},

"requires\_HITL":{"type":"boolean"},

"notes":{"type":"string"}

},

"required":["sufficient","trust","fallback\_needed","requires\_HITL"]

}

---

22) What’s Ready to SHIP now

Fast‑path Gemma with TTFT ≤ 800 ms target.

Chroma + Neo4j offline retrieval with Obsidian pipeline.

MinIO artifact packaging with .integpkg.

Langfuse + Phoenix wiring (spans/metrics).

BLOCKERS to watch:

GPU commissioning (RAVEN) + model kernels.

HITL panel + Keycloak hardening.

Online access gating (kept OFF by default).

---

23) Final Notes for the Business GPT

Keep responses concise by default (user preference) but provide full detail when asked (this doc).

Never promise background work or future delivery; perform all work synchronously.

When ambiguous, choose offline sources, declare uncertainty, and mark with DECISION.

IntegAI generic agents :

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).