# Adam — Project Guidelines & Living Docs

**Owner:** Mike (Artist, solo developer)  
**Assistant:** ChatGPT (this chat)  
**Initialized:** Sep 4, 2025

## Purpose

A single, living document we keep up‑to‑date from this chat: project guidelines, current goals, decisions, and an index to other Adam docs. You ask questions here; I answer and—when needed—update these sections.

## Collaboration Ground Rules (for this project)

* **No‑code chat:** You can ask anything; I won’t give code patches unless you explicitly ask.
* **If/when code changes are requested:**
  + Keep instructions to **2 steps** per change.
  + **Use line numbers**, and when editing functions show the **line before and after** (context) as well as the **def before/after**.
  + **Do not include imports inside code blocks** (remind separately).
  + Assume you are **new to coding**; avoid jargon and explain intent plainly.
  + You may change how you receive info, but I’ll still include **all crucial details** so you don’t miss anything.

## Response Style (brief-by-default)

* **Default replies:** 2–5 short bullets with only the directly relevant info (answer, decision, next step).
* **Silent doc updates:** I’ll update goals/guidelines/decisions and the changelog in the background. **Do not display the living doc by default.** Only show updated/new sections when you ask (e.g., “show updates”).
* **Always surface:** privacy/safety risks, cost impacts, or irreversible changes.
* **Expand on demand:** say “details”, “why?”, “show updates”, or “full”. To change back: “verbose on/off”.

## Project Vision (concise)

Adam is a **self‑hosted‑first personal assistant** that can plan, retrieve knowledge from your own docs, and run helpful workflows—with **strong cost/privacy guardrails**. (Full brief: adam\_project\_vision.md)

### Constraints & Principles

* **Self‑host first, API smart**
* **Cost safety** (bank‑check before expensive runs)
* **Small, steady steps**
* **Observability** (log runs, inputs/outputs, decisions)
* **Privacy** (no scraping behind logins; secrets in .env)

### Target Architecture (MVP → scalable)

* **Frontend:** Next.js (web), React Native (Expo) mobile
* **Backend:** FastAPI *or* Next.js API routes
* **Data:** Supabase (Postgres + Auth + Storage) + **pgvector**
* **Search/RAG:** chunked site/docs; **citations required**
* **LLM Router:** primary = streaming API; fallback = local model; burst = cloud GPU behind bank‑check
* **Orchestration:** n8n
* **Observability:** Sentry + structured logs
* **Docs for agents:** AGENTS.md (rules/tools/coding style)

### Interfaces (high‑level)

/ask, /ingest, /tasks, /eval/run, /admin/health (illustrative; exact routes can change)

### Success Metrics (MVP)

* **RAG quality:** ≥80% “useful w/ correct citation” on eval set
* **Latency:** web answers < 6s P95; local fallback < 12s P95
* **Reliability:** >99% API uptime; <1% 7‑day error rate
* **Cost:** stay within monthly budget; **zero surprise GPU spend**

## Current Goals

1. Bring up **web/** (Next.js) with admin dashboard: auth, /ask with answers + citations.
2. Bring up **api/** with RAG (pgvector) and a simple LLM router (primary API + local fallback).
3. Wire **n8n**: nightly Pin digest, weekly index refresh, tech‑watch placeholder.
4. Add **/eval/run** (small Q/A set, log metrics).
5. Write/commit **AGENTS.md** (rules, tools, style, deploy steps).
6. Evaluate agentic RAG as a sidecar (FastAPI service) called via existing adam\_toolcall HTTP plugin; return {answer, citations, trace}. (0.5–1 day)
7. Add drop‑in plugin wrapper agentic\_rag that calls the sidecar; no edits to Adam.py. Keep pgvector now; optionally support Weaviate via adapter.
8. Define a benchmark to compare baseline RAG vs agentic RAG sidecar (quality with citations, latency P95, and cost); produce a 1‑page summary. (0.5–1 day)
9. Draft **Plugin Architecture Guidelines** (discovery, config, logging, versioning) to keep Adam.py thin. (0.5 day)
10. **Memory MVP:** Implement STM summary + pgvector LTM plugin/sidecar with write/search/summarize/consolidate ops.
11. **Memory Policies:** Add PII redaction, retention/decay rules, and a memory export/erase flow.
12. **Baseline bake‑off:** Evaluate 2–3 open‑source assistants against acceptance tests; timebox 1 day; pick one baseline.
13. **MVP Alignment Contract:** Draft 1‑page spec (goals, non‑goals, demo scenarios, acceptance tests, kill switches) and commit to repo.

**Self‑Improvement roadmap priorities**: - Step 1: Router rule (API vs local)  
- Step 2: Version the system prompt  
- Step 3: Create a small eval set (10–20)  
- Step 4: Define 5 feedback tags (too\_long, off\_topic, wrong\_tone, hallucinated, great)  
- Step 5: Proposal workflow (Observe → Propose → Simulate → Report → Apply)  
- Step 6: RAG starter pack (folder with key notes/docs)  
- Step 7: Monthly prompt‑evolution on eval set  
- Step 8: Guardrails (privacy defaults, fallback, reversible changes)

## Engineering Notes

* **Lean core (Adam.py)**: Keep as condensed as possible; prefer plugins/sidecars; current file is >3.5k lines and hard to manage. We will avoid adding new code directly to Adam.py and favor single-file plugins or sidecar services.

$1- **2025‑09‑12 — Decision:** Confirmed **alignment-first plan**: use an open‑source assistant from GitHub as the base; run a 1‑day bake‑off with acceptance tests; adopt plugins/sidecars; keep Adam.py lean; use kill switches to avoid time sink.

* **2025‑09‑08 — Decision:** Add **Startup Self‑Audit + Adaptation** via sidecar/plugin. Detect HW/SW changes, plan safe config/runtime adaptations, and gate any code proposals behind eval + approval. Keep Adam.py untouched.
* **2025‑09‑05 — Article Review:** *Advanced agent with summarized STM + vector LTM (Marktechpost, 2025‑09‑02)* — **Relevance:** 3, **Impact:** +2, **Effort:** M (policy+eval) / L (scale). **Decision:** Adopt a two‑tier memory design (STM summary + pgvector LTM) via plugin/sidecar; add eval and policies.
* **2025‑09‑04 — Summary:** Established article triage workflow; reviewed Elysia (agentic RAG); agreed to prefer sidecars/plugins and keep Adam.py condensed; planned sidecar eval and plugin wrapper. Added goals to benchmark sidecar vs baseline and to draft plugin architecture guidelines.
* **2025‑09‑04 — Article Review:** *Elysia (agentic RAG with decision trees, DSPy; Weaviate‑centric)* — **Relevance:** 3, **Impact:** +2, **Effort:** S (eval)/L (integrate), **Decision:** Add to docs + Backlog 0.5–1d eval, **Next:** run demo & jot design note. Link: marktechpost + weaviate blog.
* **2025‑09‑04** — Document initialized. Imported current vision, priorities, and collaboration rules. — *ChatGPT*

## Architecture Guide — Agent Memory (MVP)

**Why:** The referenced tutorial shows pairing a summarized short‑term memory with a vector‑based long‑term memory. This aligns with Adam’s goals and can be done as a plugin/sidecar without touching Adam.py.  
**Refs:** Marktechpost tutorial (Sep 2, 2025).

### Memory tiers

* **Short‑Term (STM):** Rolling window of last N messages + a **running summary** that compresses older turns once token use crosses a threshold.
* **Long‑Term (LTM):** **pgvector** table storing compacted facts/events with metadata for recency/importance and source, retrievable via similarity search.

### Data model (LTM)

memories(id, text, embedding, importance INT, created\_at, last\_seen\_at, source, tags TEXT[], pinned BOOL, pii BOOL)

### Core operations

* **write(event):** decide if it’s memory‑worthy (rule: explicit “remember this”, or classifier). Store text + embedding; set importance.
* **summarize(history):** periodically compress older context into STM summary.
* **search(query):** k‑NN over embeddings; **rerank** by recency\*α + importance\*β.
* **consolidate():** nightly merge duplicates, drop low‑value items (decay), and pin critical facts.

### Policies

* **PII:** redact or mark pii=true; never expose by default; add export/erase endpoint.
* **Cost/Latency:** cap STM tokens; batch embeddings; consolidate off‑peak.
* **Quality:** require **citations** when LTM items originate from documents.

### Interfaces (plugin/sidecar)

* memory.write(text, tags?, source?) → id
* memory.search(query, k=8) → [items]
* memory.summarize(history) → summary
* memory.consolidate() → report

### Evaluation (mini‑suite)

* **Recall:** seed 10 facts; ask 10 questions; target ≥80% correct with citations.
* **Latency:** P95 search+rerank < 150ms on 10k items.
* **Cost:** ≤ $0.50 for the eval run.

## MVP Alignment Contract (to avoid wasted time)

* **Base from GitHub:** pick + fork an open-source assistant as our baseline (MIT/Apache) and extend via plugins/sidecars.
* **Timebox:** 1-day bake‑off across 2–3 candidates; stop after 1 day.
* **Acceptance tests:** (A) RAG with citations on our eval set, (B) plugin/HTTP tool hook working, (C) admin observability (logs of prompts/answers).
* **Out of scope (MVP):** multi-tenant, fine-tuning, complex UI; defer.
* **Kill switches:** If any candidate fails ≥2 tests or setup exceeds 2 hours, drop it.

## Baseline Bake‑off Checklist

* Spin up: AnythingLLM, Open WebUI, Dify (or 2 of 3 if time is tight).
* Run acceptance tests and capture numbers: answer quality (with citations), P95 latency, effort to add a tool/plugin.
* Choose one baseline and create a short decision note.

## Documentation Index (local project)

* **Vision brief:** /mnt/data/adam\_project\_vision.md
* **Self‑improvement roadmap:** /mnt/data/adam\_self\_improvement\_roadmap.txt
* **Project tree snapshot:** /mnt/data/project\_tree.txt
* **Main app script (reference only):** /mnt/data/Adam.py

Ask for any of these to be summarized, expanded, or turned into a checklist.

## Article Intake & Review

This is how we’ll evaluate anything you paste here (articles, videos, tweets, repos). I’ll reply with a short **Review Card** and update docs if it’s a keeper.

### Triage checklist (what I’ll answer for every item)

1. **Relevance (0–3):** 0 = off-topic, 1 = tangential, 2 = useful adjacent, 3 = core to Adam.
2. **Potential impact (−2…+3):** −2 = risky/harmful, 0 = neutral, +1 = small win, +2 = solid win, +3 = game‑changer.
3. **Effort (S/M/L/XL):** S ≤ half‑day, M ≈ 1–2 days, L ≈ 3–5 days, XL > 1 week. (Estimate range, not a promise.)
4. **Confidence (Low/Med/High):** How sure we are about the above.
5. **Category (pick one or more):** Feature idea • RAG/data ingest • Prompting/policies • Infra/cost • UX • Security/Privacy • Dev tooling • Research watchlist.
6. **Where it fits:** Vision, Current Goals, Roadmap, or Reference only.
7. **Decision:** Add to backlog • Add to docs • Note only • Skip.
8. **Next step (if any):** 1‑liner action.

### Review Card (what I’ll post back to you)

**Title:** **Link:** **Relevance:** 0–3  
**Impact:** −2…+3  
**Effort:** S/M/L/XL  
**Confidence:** Low/Med/High  
**Category:** …  
**Where it fits:** …  
**Summary (3 bullets):** - … - … - … **Decision:** …  
**Next step:** …

### Decision rules (so it’s predictable)

* **Add to backlog** if Relevance ≥2 and Impact ≥+1 and Effort ≤L.
* **Add to docs (reference)** if Relevance ≥1 and Impact ≥0 and it clarifies our approach.
* **Research watchlist** if Relevance 1–2 but Confidence is Low; we’ll revisit monthly.
* **Skip** if Relevance 0 or Impact ≤−1.

### What happens when it’s a keeper

* I’ll add a concise entry in **Decisions & Q&A Log** and, if appropriate, append to **Current Goals** or **Roadmap** with a date-stamped note.

## Pending Questions

*(Add questions you want answered soon; I’ll keep this prioritized.)* - —

$1- **2025‑09‑12:** Added **MVP Alignment Contract** + **Baseline Bake‑off Checklist**; appended two goals; logged the alignment decision. - **2025‑09‑08:** Set rule to keep the living doc hidden by default and only display updated/new sections on request. - **2025‑09‑08:** Logged decision to add **Startup Self‑Audit + Adaptation** in Decisions & Q&A. - **2025‑09‑05:** Added “Architecture Guide — Agent Memory (MVP)”, appended Memory MVP and Policies to Current Goals, and logged decision from article review.

* **2025‑09‑04:** Summarized session; added goals for benchmarking and plugin architecture guidelines.
* **2025‑09‑04:** Added objectives for agentic RAG sidecar/plugin and noted goal to keep Adam.py condensed (>3.5k lines).
* **2025‑09‑04:** Logged review & decision for Elysia (agentic RAG) in Q&A Log.
* **2025‑09‑04:** Created this living document and seeded it with current materials.

## How to Use This Chat to Update Docs

* **Ask anything** (e.g., “What should the MVP include?”). I’ll answer plainly.
* If the answer should live in the docs, I’ll add it to **Decisions & Q&A Log** or **Guidelines** automatically and note the change in **Changelog**.
* You can also say things like:
  + “**Add to guidelines:** Prefer self‑hosted vector DB unless costs spike.”
  + “**Log a decision:** Use Next.js API routes instead of FastAPI for MVP.”
  + “**Summarize & update:** Pull today’s answers into the docs.”