Reflect Memory Architecture

Date Updated: May 21, 2026

Reflect Memory sits **below the AI tools** (ChatGPT, Claude, Cursor, Gemini, Grok, n8n, and the rest) and provides a persistent, user- and team-scoped memory service. Every write is explicit, every read respects visibility controls, and a deterministic audit trail streams through the same stack used for production workloads.

System Layers

- **Users & Dashboard** – Clerk-backed UI, service-key + JWT auth, multi-factor dashboards per tenant.
- **REST API** – Fastify, `openapi.json`, `/memories/*`, `/agent/*`, `/query`, `/mcp`, `/admin` routes, per-operation rate limiting (100/min).
- **Memory Service** – `better-sqlite3` / eventual Postgres, strict JSON schema, explicit `user_id` required for every function.
- **MCP Server** – Express on port 3001, proxied at `/mcp`. Tools: `read_memories`, `write_memory`, `browse_memories`, `search_memories`, `get_memories_by_tag`, `get_memory_by_id`, `get_latest_memory`, `read_team_memories`, `share_memory`. Vendor resolution happens per request via `RM_AGENT_KEY_<VENDOR>`.
- **Graph helpers** – `memory-graph.ts`, `memory-briefing.ts`, `get_graph_around`, and the future `get_current_state(topic)` tooling keep temporal context, supersession, and open threads accessible to agents.

Data Model & Temporal Guarantees

Field	Purpose
`id`	UUIDv4 primary key
`user_id`	Ownership; enforced by service layer, never bypassed
`content`	Up to 100KB text, with `tags`, `allowed_vendors`, `origin`, `created_at`, `updated_at`, `deleted_at`
`allowed_vendors`	Visibility guard for ChatGPT, Claude, Cursor, etc. `["*"]` = all tools, otherwise scoped list

Every read path adds `AND deleted_at IS NULL` plus vendor visibility checks. Team sharing flows (`share_memory`, `read_team_memories`) layer an additional scope for teammates.

Transport & Integration Surface

- **REST**: Handles dashboard, CLI, integrations, HTTP clients. Schema validation + OpenAPI spec at `/openapi.json` and `/docs`.
- **MCP**: Streamable HTTP sessions, vendor-scoped context, per-connection session metadata, supports `get_latest_memory`, `read_team_memories`, `share_memory`.
- **Agent keys**: `RM_AGENT_KEY_CHATGPT`, `RM_AGENT_KEY_CLAUDE`, etc. `resolveVendor()` iterates them with timing-safe compare, then closes over the vendor name so tools cannot spoof origin.
- **OAuth**: Claude OAuth flow (`/chatgpt/authorize` etc.) uses `RM_PUBLIC_URL` to build redirect URIs. When no agent keys are provided, MCP returns 404 until `RM_AGENT_KEY_*` or `RM_PUBLIC_URL` enables it.

Deployment Modes & Boundary Controls

Configuration is centralized in `resolveDeploymentConfig`, keyed on `RM_DEPLOYMENT_MODE`, `RM_DISABLE_MODEL_EGRESS`, `RM_REQUIRE_INTERNAL_MODEL_BASE_URL`, `RM_ALLOW_PUBLIC_WEBHOOKS`, and `RM_ALLOWED_MODEL_HOSTS`.

- `hosted` – multi-tenant (default), network boundary public, webhook exposure allowed, model egress optional.
- `isolated-hosted` – dedicated runtime and DB per tenant, networks can be locked, egress still controlled.

- `self-host` – private network boundary, `disableModelEgress` defaults to true, `requireInternalModelBaseUrl` defaults to true, `allowedModelHosts` must list internal model endpoints, no public webhooks.

SSO config (`RM_SSO_ENABLED` , `RM_SSO_JWKS_URL` , `RM_SSO_ISSUER` , `RM_SSO_AUDIENCE` , `RM_SSO_EMAIL_CLAIM`) is validated at startup. Model host policy ensures `OLLAMA` , `llama.cpp` , or enterprise LLM URLs are explicitly whitelisted.

Security & Compliance

- Timing-safe comparisons for every API key (`RM_API_KEY` , service keys, agent keys, per-user keys). Agent requests limited to MCP + query surfaces.
- Audit trail logs every read, write, auth event, and admin action. The `usage_events` table feeds Stripe billing logic and compliance exports.
- Telemetry is optional; self-host never phones home. Cloud deployments still ship minimal service quality metrics.
- Soft delete, `deleted_at` , and per-tenant audit table ensure recoverability plus regulatory audit readiness.

Graph, Temporal, & Async Diligence Ready

`memory-graph` tracks parent/child edges and supersession metadata so agents can answer “What changed?”, “What’s open?”, and “What superseded this?” without piecing it together manually. `get_graph_around` already powers tooling; the next five MCP helpers (`get_open_tickets` , `get_unresolved_threads` , `get_recent_decisions` , `get_superseded_decisions` , `get_current_state(topic)`) will make operational status deterministic.

Roadmap Notes

- **Postgres migration** – `schema-postgres.sql` ready; swaps to `pg` / `postgres.js` , adds JSONB, `tsvector` search, usage partitions.
- **Usage metering toggle** – Stripe integration wired; gating just waiting for public beta flag flip so quotas react.
- **Semantic search** – plan to layer vector embeddings on top of `tsvector` for meaning-based recall.
- **Async diligence content** – a markdown + prompt-first docs bundle ensures every customer, investor, and AI can self-serve the technical Q&A.

Current State & Contact

- API: `https://api.reflectmemory.com` (Reflect Memory cloud, self-managed infrastructure — not Railway or Vercel).
- Dashboard: `https://reflectmemory.com` (same stack; GitHub Actions deploy to dedicated hosts).
- Deployments: Hosted cloud, isolated private deploys, and self-host pilots in progress with healthcare technology and enterprise SaaS customers.
- Integrations: ChatGPT, Claude, Cursor, Gemini, Grok, n8n (OpenAPI + MCP).
- Contact: `Tamer Shafik <ts@reflectmemory.com>` for architecture/deployment; `Van Mendoza <vm@reflectmemory.com>` for GTM/positioning.

Reflect Memory is **not an app**, it is **the memory substrate** that AI-native teams plug into. The async diligence bundle (this doc + downloads + prompts) proves the claim: the docs you need are ready for your AI to ingest before you ever schedule another call.