Built Like Enterprise Infrastructure

A typed Java runtime, a standard PostgreSQL data layer, and a verifiable supply chain — GoodMem's trust posture for production

In the enterprise, the buyer carries the risk. Most of the tooling in AI memory and agent infrastructure was built to prototype quickly — Python scripts and notebooks, persisting state in whatever store was handy, distributed without provenance or a supply chain anyone can audit — and it fails the first serious security review. **GoodMem** is built the other way: as infrastructure. A typed **Java** service on the JVM, persisting every byte in a standard **PostgreSQL** database you control, shipped as **distroless**, signed and verified container images with **SLSA Level 3** provenance, FIPS-validated cryptography, a software bill of materials, and a closed, vetted contributor model. It is **ISO 27001 certified**, with a SOC 2 Type II audit in its final stage. For the conservative decision-maker whose job is to say no to risk, that is the difference between an experiment and a system you can deploy.

Java 21

Engineered runtime
a typed JVM server — not a Python prototype

PostgreSQL

Open, portable data layer
your data in a standard DB — no proprietary store, no lock-in

ISO 27001

Independently certified
third-party audited · SOC 2 Type II in final stage

An enterprise backbone, not a prototype

A production system has two load-bearing parts — the runtime that executes your logic and the store that holds your data. GoodMem builds both on technology enterprises already trust, operate, and audit.

Engineered in Java, not scripted in Python

The tools that dominate this space began as fast-moving developer frameworks: Python (LangGraph, LlamaIndex) or Node.js (Flowise), excellent for experimentation. GoodMem's server is a typed Java service on the JVM — the runtime that has carried regulated, high-volume production systems for two decades, with the concurrency, memory safety, observability, and operational tooling that prototyping never had to build. The model layer can be Python; the secured, operable backbone should not be.

Your data in PostgreSQL, not a black box

Where your data sits is the first thing a security review asks. GoodMem keeps everything — memories, text, vector embeddings, page images, even the job queue and audit logs — in one PostgreSQL database you run and control. No proprietary format, and no separate vector database bolted on: embeddings live in Postgres via pgvector, so there is one system to secure, back up, and audit — not a sprawl of Redis, Elasticsearch, and a proprietary index, each with its own license and attack surface. Standard SQL, backups, and encryption — your data stays portable, inspectable, and yours.

Security & supply-chain control	GoodMem	LangGraph	LlamaIndex	Flowise
Engineered runtime	**Java 21 · JVM**	Python	Python	Node.js
Standard, open data store	**PostgreSQL**	BYO	BYO	BYO
Distroless, non-root images	**Yes**	No	No image	No
SLSA L3 build provenance	**Yes**	None	None	None
Release-gated CVE scanning	**Gated**	Version bumps	Version bumps	Version bumps
SBOM + license compliance	**Yes**	No	No	No

Secured and verifiable

GoodMem treats the software supply chain as a first-class security surface. Release images are distroless and non-root — no shell, no package manager, nothing for an attacker to pivot to — cryptographically signed and checked against a pinned digest before they run, with SLSA Level 3 provenance you can verify and a software bill of materials for every build. Every pull request and release is gated by dependency vulnerability scanning across multiple independent databases — a release cannot ship with an unresolved medium-or-higher CVE — and by release-blocking license and attribution compliance. Cryptography uses a FIPS-validated library in approved-only mode, contributions come only from a vetted internal team behind a closed repository, and the program is independently audited: ISO 27001 certified, with a SOC 2 Type II audit in its final stage.

Operable and integrable

GoodMem is built to run where your other production services run. It is self-hostable inside your boundary, exposes a native Prometheus metrics endpoint (JVM, HTTP, gRPC, and PostgreSQL) and Kubernetes-ready liveness, readiness, and startup probes alongside the standard gRPC health protocol, and correlates every request across REST and gRPC, with a durable per-request retrieval audit trail. Integration is standards-based: a protobuf-defined gRPC API and a parallel REST API with a live OpenAPI specification, a built-in MCP server for agents, and pre-built, published client SDKs for Python, Java, and more. TLS runs with your own certificates, self-signed, or automated issuance.

Memory without model lock-in

An agent's memory is an asset that compounds: the longer a system runs, the more it knows about your data, your metrics, and your pitfalls. Where that asset lives decides how much freedom you keep. Most memory products place a language model inside the write path itself: each new memory is produced by calling a configured model that decides what to keep and how to phrase it. Every write then depends on a model endpoint, and what accumulates in the store is that model's paraphrase of your content.

GoodMem does not. Its **write path is chunking, embedding, and indexing**; its read path is vector search and reranking. What enters the store is your source text, with its page images, and a language model appears at exactly one point — an **optional summarization step** at retrieval time that can be turned off. Each model slot is a configuration choice. Embeddings can come from OpenAI, Cohere, Jina, or Voyage, or from TEI and vLLM servers on your own hardware; reranking runs on Cohere, Jina, Voyage, or the same self-hosted engines; the optional summarizer works against OpenAI, OpenRouter, or a local vLLM, Ollama, or llama.cpp endpoint. A deployment that sends nothing to an outside AI vendor is a supported configuration, and the agent on the other side connects over MCP, gRPC, or REST, whatever model it runs on.

Switching models is therefore a configuration change, not a migration. The memories themselves are **rows in your PostgreSQL database**, stored with the original text and page images from which every vector was derived, so the corpus is never reduced to a single model's vector space. Change the agent's model, or the embedder behind the index, and nothing the system has learned is lost. The vendors compete for a slot in your stack; the memory stays where it was.

Trust as the foundation

PAIR Systems builds GoodMem as enterprise infrastructure — engineered, secured, and certified to the standard regulated industries require. The result is memory you can trust on a foundation you can audit, monitor, and operate like any other production system you run.

Methodology. Security comparison reflects a June 2026 review of public documentation, repositories, and distribution artifacts for the named tools (LangGraph / LangChain, LlamaIndex, Flowise) versus GoodMem, comparing the security and supply-chain posture of each project's distributed software, not company-level certifications; "BYO" denotes persistence the integrator must assemble. GoodMem's controls are verified against its source and build pipeline: a Java 21 / Gradle server; a single PostgreSQL data layer holding all state, including vector embeddings via pgvector; distroless, non-root release images, cosign-verified with sha-256 digest pinning; SLSA Level 3 build provenance; release-gating dependency vulnerability scanning (OSV-Scanner and Grype) across every pull request and release, save a small set of documented, time-bounded exceptions; a FIPS-validated cryptographic library in approved-only mode; SBOM generation; release-blocking license and attribution compliance; a Prometheus metrics endpoint; Kubernetes liveness/readiness/startup probes and the gRPC health protocol; and gRPC, REST/OpenAPI, and MCP interfaces with published client SDKs. Vulnerability scanning covers third-party dependencies; the per-request audit trail covers retrieval. The write-path comparison reflects a July 2026 review of public documentation for widely used memory layers (Mem0, Zep, Letta), each of which invokes a configured language model to extract or consolidate memories at write time; GoodMem's write and read paths and its provider integrations (OpenAI, Cohere, Jina, Voyage, TEI, vLLM, Ollama, llama.cpp, OpenRouter, LiteLLM) are verified against the server source. GoodMem is ISO 27001 certified; a SOC 2 Type II audit is in its final stage (observation period complete). Role-based access control is on the roadmap.