Cogeto: A Verifiable, Sovereignty-First Memory Architecture for Large Language Model Assistants

Design, Mechanisms, and Rationale of a Private, EU-Hosted System for Correctable Long-Term Memory, Consolidation, and Provable Deletion

Ivan Golubić
MVT Solutions Group d.o.o., Croatia, European Union
MCTO Advisory d.o.o., Croatia, European Union

info@mvt-solutions.com | hi@mctoadvisory.com | ivan@themrcto.com
Project domain: cogeto.eu

Working paper. The system described is under active development.
July 2026

Abstract

Large language model (LLM) assistants are increasingly entrusted with the working context of professional life: decisions, commitments, relationships, and open obligations scattered across email, calendars, notes, and documents. Yet the memory layers attached to such assistants are typically opaque accumulators. They remember, but they cannot show what they remember, why they believe it, whether it is still true, or whether it was ever really deleted. For privacy-sensitive users, and for organizations operating under the European Union's data protection and artificial intelligence regulations, this opacity is disqualifying. In this working paper we propose Cogeto, a private, EU-hosted memory architecture for LLM assistants in which every trust claim is backed by an inspectable artifact. Cogeto models memory as discrete, source-linked facts carrying an explicit quality status drawn from a closed seven-state vocabulary, a visibility scope, and a temporal validity interval. Facts enter memory only through a pipeline that includes an independent verification pass; they age, contradict, and supersede one another under a nightly consolidation process we call dreaming, by analogy with sleep-dependent memory consolidation in biological systems; they are retrieved through hybrid, filtered, rank-fused search in which visibility is a hard precondition rather than a ranking signal; and they are deleted through a compensating transaction protocol that terminates in a cryptographically hash-chained deletion receipt, giving practical operational form to the right to erasure. The architecture separates a synchronous fast path from an asynchronous slow path, isolates all model access behind a swappable gateway that supports a European API provider by default and fully isolated local models where required, and pseudonymizes sensitive entities locally before any external model call. We describe the system's design principles, mechanisms, evaluation methodology, and regulatory alignment, and we argue that verifiability, not capacity, is the property that makes machine memory trustworthy.

Keywords: large language models, long-term memory, retrieval-augmented generation, memory consolidation, data sovereignty, right to erasure, verifiable systems, human oversight, hybrid retrieval, temporal data

1. Introduction

1.1 Motivation

The adoption of artificial intelligence in organizations has moved from experiment to default. The Stanford AI Index reports that 78 percent of organizations used AI in 2024, up from 55 percent a year earlier, with generative AI use in at least one business function more than doubling over the same period [40]. At the same time, the trust required to make such systems genuinely useful has not kept pace with their capability. In a 2024 benchmark study of 2,600 privacy and security professionals, more than one in four organizations reported having banned generative AI tools at least temporarily over privacy and data security concerns, while nearly half admitted that non-public company information had been entered into such tools [41]. The tension is structural: the assistants that would be most valuable are precisely those that remember the most sensitive material, and the more they remember, the less their users are able to see, correct, or revoke.

This tension is sharpest for a specific population: independent professionals and small advisory teams whose working context is, in substance, other people's confidential information. Consultants, counsel, financial advisors, and fractional executives accumulate decisions, promises, and obligations across email threads, calendar entries, and notes. A system that could answer the simple question "what did I decide, promise, and commit to, and what is still open?" would be transformative for them. But such users cannot, professionally or often legally, submit that context to an opaque, foreign-operated service whose memory can be neither inspected nor provably erased. In the European Union, this is not merely a preference. The General Data Protection Regulation establishes a right to erasure and mandates data protection by design and by default [34], with cumulative enforcement fines measured in the billions of euros [42], and the Artificial Intelligence Act introduces transparency obligations for AI systems [35]. The European Commission's data strategy explicitly frames control over data infrastructure as a matter of European sovereignty [36].

We take this convergence of practical need and regulatory demand as a design brief. The question this paper addresses is not how to make an LLM assistant remember more, but how to make what it remembers worthy of trust.

1.2 Position: from remembered to verifiable

Our central claim is that trust in machine memory cannot be asserted; it must be evidenced. A memory system earns trust when each of its trust claims corresponds to an artifact the user can inspect. We adopt this as an explicit design rule: **if a trust claim has no artifact, it is marketing; if it has one, it is a mechanism.** Concretely, in the system we propose:

- The claim "I remember accurately" is evidenced by a published evaluation score computed against a hand-labeled corpus on every release.
- The claim "I know where each fact came from" is evidenced by mandatory, non-nullable provenance on every stored memory, surfaced as a source link in the interface.
- The claim "I know when facts stop being true" is evidenced by temporal validity intervals and an explicit quality status on every fact, maintained by an autonomous consolidation process.
- The claim "I forget when told to" is evidenced by a signed, hash-chained deletion receipt produced by a compensating transaction protocol that removes the original bytes, the derived facts, and their vector representations, and then verifies the absence of remnants.

- The claim "I act only with permission" is evidenced by a server-side approval state machine whose every transition is audit-logged.
- The claim "your data does not leave your jurisdiction" is evidenced by single-tenant deployment on European infrastructure, a European primary model provider, optional local pseudonymization of sensitive entities before any external call, and the architectural ability to route all model traffic to fully isolated local models.

We refer to the resulting property as verifiable memory. It is, to our knowledge, a combination not offered by memory systems currently attached to LLM assistants, which overwhelmingly optimize for capacity and recall rather than inspectability, correctability, and provable forgetting.

1.3 Contributions

This working paper makes the following contributions:

- A conceptual model of assistant memory as discrete, source-linked facts with a closed seven-state quality vocabulary, explicit visibility scopes, and bi-temporal validity, designed so that correctness claims are checkable rather than assumed (Section 4).
- An architecture that enforces the model's invariants structurally: a modular monolith with domain-driven boundaries, a strict separation of a synchronous retrieval path from an asynchronous maintenance path, and a single source of truth from which all derived indices are rebuildable (Section 5).
- An ingestion pipeline with an independent verification pass, in which no extracted fact is trusted until a second, independently prompted check confirms that its cited source actually supports it (Section 6).
- A consolidation process, dreaming, that performs deduplication, contradiction detection, supersession, and staleness marking as an autonomous nightly cycle, grounded by analogy in the neuroscience of sleep-dependent memory consolidation, and surfaced to the user as a daily, fully traceable digest (Section 7).
- A retrieval design that fuses semantic, lexical, and entity signals with reciprocal rank fusion while treating visibility and sensitivity as hard preconditions evaluated inside the store, never as score adjustments (Section 9).
- A provable deletion protocol combining the saga pattern with hash-chained, signed receipts and autonomous remnant verification, giving operational form to the right to erasure (Section 10).
- A model access layer that is European by default and isolable by construction: a swappable gateway routing to a European API provider, a local pseudonymization tier for sensitive entities, and full support for locally hosted models such that no content need ever leave the deployment boundary (Section 12).
- An evaluation methodology in which measured extraction quality is a continuous integration gate and a published artifact, not an internal metric (Section 14).

1.4 Scope and status

Cogeto is a system under active development by the authors; this paper describes its design and rationale rather than a completed evaluation. Where we report mechanisms, they are specified and, in part, implemented; where we report quality thresholds, they are commitments the evaluation harness is being

built to enforce. We deliberately exclude commercial considerations from this paper and confine ourselves to architecture, mechanisms, and the regulatory context that motivates them.

2. Background and Related Work

2.1 Large language models and retrieval augmentation

Modern assistants are built on Transformer-based language models [1] whose few-shot competence at scale was established by GPT-3 [2]. Such models are, however, parametric: their knowledge is frozen at training time, is not attributable to sources, and cannot be selectively corrected or deleted. Retrieval-augmented generation (RAG) addresses these limits by conditioning generation on documents retrieved at inference time [3], with dense passage retrieval providing the learned-embedding retrieval substrate [4] and retrieval-integrated pretraining demonstrating the value of making retrieval intrinsic to the model [5]. Surveys of the rapidly growing RAG literature catalogue its variants and its central promise: grounding generation in inspectable, updatable external knowledge [6].

Cogeto is a RAG system in the broad sense, but it inverts the usual emphasis. Most RAG work treats the external store as a passive corpus and concentrates innovation in retrieval and prompting. We treat the store itself as the primary object of engineering: what enters it, under what verification, with what metadata, how it ages, and how it is provably removed.

2.2 Memory for LLM agents

A distinct line of work equips LLM agents with explicit memory. MemGPT frames the model's context window as a scarce resource managed like virtual memory, paging information between fast and slow tiers [9]. Generative agents demonstrated that long-horizon believable behavior requires a memory stream combined with periodic reflection, in which higher-level conclusions are synthesized from accumulated observations [10]. Recent surveys organize the design space of agent memory along acquisition, management, and utilization dimensions [11]. These systems establish that memory management, not merely memory capacity, is the operative problem. Our work adopts that lesson and extends it in a direction the agent-memory literature has largely not pursued: memory as an auditable, correctable, legally accountable artifact with explicit quality states and provable deletion.

2.3 Hallucination and verification

LLMs generate fluent falsehoods. Surveys of hallucination in natural language generation [7] and in large language models specifically [8] document the phenomenon's breadth and taxonomize its causes. Mitigations relevant to our design include Chain-of-Verification, in which a model drafts verification questions about its own output and answers them independently before finalizing [12]; SelfCheckGPT, which detects hallucination through sampling-based consistency without external resources [13]; and FActScore, which decomposes long-form generations into atomic facts and verifies each against a source [14]. Cogeto internalizes this literature at the point of memory formation: extraction output is treated as a claim, not a fact, until an independently prompted verification pass confirms that the cited source supports it, and the atomic-fact granularity of storage is chosen precisely so that verification has a well-defined object.

2.4 Hybrid retrieval and rank fusion

Lexical retrieval in the probabilistic relevance framework, exemplified by BM25 [16], and approximate nearest-neighbor search over dense embeddings, exemplified by Hierarchical Navigable Small World graphs [17], have complementary failure modes: the former misses paraphrase, the latter misses exact identifiers, names, and codes. Reciprocal rank fusion is a simple, robust method for combining ranked lists from heterogeneous retrieval systems that outperforms more elaborate fusion and learning approaches in practice [15]. Cogeto's retrieval fuses three signals, semantic, lexical, and entity-based, under reciprocal rank fusion, and contributes a policy layer above the fusion: visibility and sensitivity operate as hard preconditions inside each underlying query, and memory quality states operate as post-fusion score multipliers (Section 9).

2.5 Temporal data management

Facts are true during intervals, not forever. The temporal database literature distinguishes valid time, when a fact holds in the modeled world, from transaction time, when the database knew it, and this bi-temporal distinction was standardized in SQL:2011 [18]. Cogeto applies bi-temporal modeling to assistant memory: every fact carries a validity interval maintained independently of its ingestion timestamp, superseded facts have their intervals closed rather than their rows destroyed, and point-in-time queries over past belief states are a first-class retrieval mode (Section 8).

2.6 Consolidation in biological memory

Our consolidation process borrows its name, dreaming, from a robust body of neuroscience. Sleep actively consolidates memory: newly encoded, labile traces are reactivated, redistributed, and integrated with existing knowledge during sleep, with sleep favoring the extraction of gist and the integration of new information into schemas [19, 20, 21]. Complementary learning systems theory explains why biological cognition separates a fast-learning hippocampal store from a slow-learning neocortical one: interleaved, offline replay allows integration without catastrophic interference [22]. Catastrophic forgetting, the tendency of connectionist systems to overwrite prior knowledge when trained sequentially, is the failure mode this separation avoids [23, 24], and it is one motivation for externalizing assistant memory into a non-parametric store rather than continually fine-tuning the model. We are explicit that the analogy is architectural, not biological: Cogeto's dreaming is a scheduled batch process, not a claim about machine cognition. But the division of labor it implements, fast online encoding followed by slow offline reorganization, deliberately mirrors the biological pattern, for the same reason: reorganization is expensive and must not block behavior.

2.7 Distributed systems patterns

Cogeto's reliability mechanisms are deliberate applications of established patterns. Long-lived operations spanning multiple stores are structured as sagas, sequences of local transactions with compensating actions, following Garcia-Molina and Salem [25]; state changes and their downstream work are recorded atomically using the transactional outbox pattern [26]; the codebase is organized around domain-driven bounded contexts [27]; and the system is deliberately a modular monolith rather than a distributed microservice topology, consistent with the argument that monolith-first designs avoid premature distribution costs [28]. Tamper-evident audit structures follow the linked time-stamping construction of Haber and Stornetta [37] and the secure audit log design of Schneier and Kelsey [38].

2.8 Human oversight and privacy tooling

Guidelines for human-AI interaction emphasize efficient invocation, correction, and dismissal of machine initiative [29], and the human-in-the-loop literature surveys architectures in which human judgment gates machine action [30]. Cogeto operationalizes oversight as a server-side state machine rather than an interface convention (Section 11). For local privacy screening, mature open-source tooling exists: the Presidio framework provides configurable detection and anonymization of personally identifiable information [31], and GLiNER demonstrates that compact bidirectional transformer models can perform open-type named entity recognition efficiently on commodity CPUs [32]. European guidance on pseudonymization techniques [33] frames such screening as a recognized safeguard under data protection law.

2.9 Positioning

Table 1 summarizes how the strands above map onto the system's mechanisms; each mapping is developed in the section indicated.

Literature	What it establishes	Cogeto mechanism	Section
RAG and retrieval [3, 4, 5, 6]	External, updatable grounding beats parametric recall	The memory store as the engineered object; source-linked answers	4, 9
Agent memory [9, 10, 11]	Management, not capacity, is the problem	Status vocabulary, consolidation, governance surface	4, 7
Hallucination and verification [7, 8, 12, 13, 14]	Model output requires independent checking	Verification pass at admission; measured agreement	6, 14
Rank fusion and hybrid IR [15, 16, 17]	Heterogeneous retrievers fuse robustly by rank	Three-signal retrieval under reciprocal rank fusion	9
Temporal data [18]	Facts hold during intervals; belief has history	Bi-temporal memories, point-in-time queries	4, 8
Consolidation neuroscience [19-24]	Offline reorganization is constitutive of memory	The dreaming cycle and its digest	7
Distributed patterns [25, 26, 27, 28]	Sagas, outboxes, bounded contexts, monolith-first	Deletion saga, transactional ingestion, module rules	5, 10
Oversight and privacy [29-33]	Human gating and local pseudonymization are practicable	Approval state machine; redaction tier	11, 12
Law and policy [34, 35, 36]	Erasure, transparency, and sovereignty are obligations	Receipts, published artifacts, EU-first deployment	10, 13, 15

3. Problem Statement and Design Principles

3.1 The problem

We consider a user whose professional context is distributed across email, calendar, and notes, and who requires an assistant that can (i) answer questions about decisions, commitments, people, and open

obligations across those sources; (ii) demonstrate, for any answer, where each supporting fact came from and how current it is; (iii) accept correction, and propagate it; (iv) delete, on request, a source and everything derived from it, and prove that the deletion occurred; and (v) do all of the above without the user's data leaving infrastructure and jurisdictions they have chosen. Constraints (ii) through (v) disqualify memory designs that store raw text opaquely, mix users' data in shared indices, treat deletion as suppression, or bind the system to a single foreign inference provider.

3.2 Design principles

Six principles govern the architecture. We state them here and refer back to them throughout.

- **P1, artifact-backed trust.** Every trust claim must correspond to an inspectable artifact: a receipt, a score, a source link, a status, an interval, an audit record.
- **P2, facts, not documents.** The unit of memory is an extracted, verified, atomic fact. Raw documents are stored as originals for provenance, but they are never themselves the memory; unbounded raw text in the retrieval index is the primary cause of memory pollution.
- **P3, structural enforcement.** Invariants live in schemas, state machines, and module boundaries, not in conventions. A visibility rule that can be forgotten in one query is not a rule.
- **P4, fast path and slow path.** The user-facing path performs retrieval and answering only. Everything expensive, extraction, verification, reconciliation, consolidation, deletion, runs asynchronously and never blocks a response.
- **P5, one source of truth.** The relational store owns all facts and metadata; every derived index, including the vector index, is rebuildable from it at any time. Nothing exists only in a derived store.
- **P6, sovereignty by construction.** Deployment is single-tenant on infrastructure of the operator's choosing within the EU; the primary model provider is European; sensitive entities can be pseudonymized locally before any external call; and the model layer can be pointed entirely at local, isolated models without code change.

3.3 Non-goals

Stating what the system does not attempt clarifies what it does. Cogeto does not pursue personalization by model fine-tuning; user knowledge lives exclusively in the external store, for the reasons developed in Section 12.5. It does not attempt autonomous adjudication of conflicting information; contradiction is detected mechanically and resolved by the user, because on questions about the user's own commitments the user is the authority. It does not aim at unbounded integration breadth; the ingestion surface is deliberately confined to the three sources, correspondence, calendar, and notes, that carry a professional's commitments, on the argument that a narrow, verified memory outperforms a broad, polluted one. It does not treat conversational fluency as a goal in itself; the conversational surface is constrained to answer from retrieved, attributable facts, and an eloquent unsupported answer is scored as a failure, not a feature. And it does not claim that architecture alone constitutes regulatory compliance, which always retains organizational and contractual components; the claim, made precisely in Section 15, is that the architecture makes the technical share of compliance demonstrable.

4. A Conceptual Model of Verifiable Memory

4.1 The memory unit

The atom of the system is the memory: one durable fact, extracted from a source, carrying its complete trust metadata. Formally, a memory is a tuple comprising an owner; a scope; a provenance reference; a quality status; a validity interval; the fact content itself; and a reference to its vector representation. Three of these fields are constrained in ways that carry the design.

Provenance is mandatory and non-nullable. Every memory references the exact source that produced it: an email, a calendar event, a note, a document, or, importantly, the user's own statement in conversation, which is provenance too. There is no such thing as an orphan fact. This single schema constraint is what makes the interface's promise, click any fact and see where it came from, universally true rather than usually true, and it is what makes deletion cascades computable: because every derived fact knows its source, removing a source can provably remove everything derived from it.

Scope is a closed enumeration, not free metadata. A memory is either private, visible only to its owner, or shared, visible to members of the same organization. Scope participates in every read path as a hard precondition (Section 9); it is a property the storage layer enforces, not a filter the application remembers to apply.

Status is a closed seven-state vocabulary. Every memory is in exactly one of the following states, and the state is part of the row, not an annotation:

Status	Meaning	Who may set it
active	Current and verified; the default state a fact reaches only after passing verification	The ingestion pipeline, after the verification pass
user_approved	Explicitly confirmed by the user; the strongest state	Only the user
uncertain	Extracted but not adequately supported by its source, or flagged by consolidation as low-confidence	The verification pass or consolidation
contradicted	In conflict with another memory; both parties to the conflict are marked and linked	Only the consolidation process
outdated	Past its validity interval or judged stale by consolidation	Consolidation or the user
replaced	Superseded by a newer fact, to which it points; excluded from default retrieval but preserved as history	Consolidation, upon supersession
sensitive	Restricted; participates in retrieval only under an explicit sensitivity gate	The user or ingestion policy

The status vocabulary is deliberately closed and small. Its transitions are owned by a single domain aggregate: only the consolidation process may mark a fact contradicted, only the user may mark it user_approved, and only the deletion protocol may remove it. Ownership of transitions is what makes the states meaningful; a status anyone can set is a comment, not a state.

4.2 Time as a first-class dimension

Following bi-temporal practice [18], every memory distinguishes when it was learned (transaction time, the ingestion timestamp) from when it holds (valid time, the interval between valid_from and

valid_until). Supersession is an interval operation: when a newer fact replaces an older one, the older fact's interval is closed, its status becomes replaced, and it acquires a pointer to its successor. Nothing is destroyed. The result is that the memory store supports point-in-time queries, what did I believe about this engagement in March, and change queries, what has changed since we last spoke, as retrieval modes rather than as reconstructions. We call the user-facing form of this capability time-travel memory: the assistant's beliefs have an undo history, and the history is itself inspectable.

4.3 Why a closed model

It is worth stating why the model is deliberately restrictive: mandatory provenance, closed scopes, closed statuses, owned transitions, preserved history. Each restriction converts a soft quality goal into a checkable invariant. "We know our sources" becomes a NOT NULL constraint. "We respect visibility" becomes a precondition no query can omit. "We handle conflicting information responsibly" becomes a pair of linked rows in a specific state that a user can see and resolve. "We do not silently rewrite the past" becomes closed intervals and successor pointers. The model is the paper's first answer to its own question: memory becomes trustworthy when its correctness claims stop being prose and start being schema.

4.4 Scopes and collaborative memory

The two-scope model is minimal by intent, but the shared scope carries a specific collaboration thesis: small teams should collaborate through shared memory rather than through a shared conversation. When one member records or ingests material in the shared scope, a colleague's later question, what did we decide on this account, retrieves the shared facts with their full provenance, statuses, and history, without either party reading the other's private scope and without the team maintaining a single, ever-growing chat whose context no one governs. The design keeps the collaboration unit at the level where the trust machinery operates, the individual fact, so that everything this paper establishes about verifiability, source links, statuses, intervals, receipts, applies unchanged to collaborative knowledge. Sensitivity composes orthogonally: a fact may be shared in scope yet sensitive in handling, restricting retrieval and source access regardless of scope. Finer-grained role structures are deliberately excluded from the model at this stage; the closed two-scope vocabulary keeps the visibility gate of Section 9 simple enough to be provably ever-present, and simplicity at the gate is a security property.

5. System Architecture

5.1 A modular monolith with two processes

Cogeto is implemented as a modular monolith [28]: a single codebase organized into domain-driven bounded contexts [27], deployed as exactly two application processes. The app process serves the synchronous surface: the conversational interface, the governance dashboard, connector endpoints, and approval endpoints. The worker process executes everything asynchronous: the ingestion pipeline, verification, reconciliation, consolidation, reminders, the deletion protocol, and the execution of approved actions. Both are built from one artifact and differ only in their entrypoint.

The choice is deliberate and follows principle P3. The bounded contexts, memory, ingestion, retrieval, agents, connectors, tasks, and two seams described below, are code boundaries with three mechanically enforced rules: each module exposes exactly one public interface; no module reads or writes another module's tables; and all cross-module communication travels as domain events through a transactional

outbox. Boundary rules are verified in continuous integration, because module boundaries that nothing enforces erode. At the same time, because the modules share one process and one database, the system avoids the operational and consistency costs of premature distribution [28], which matters doubly in a deployment model where every customer runs an isolated instance: each additional network service would be multiplied across every deployment.

Two of the modules are seams: thin leaf modules that isolate an external dependency behind a stable interface. The identity seam wraps the identity provider, which answers who a user is and what organization and roles they hold; crucially, the identity provider never decides which memories a user may see, since visibility is the memory model's own logic. The model gateway seam carries every model and embedding call in the system; no other module may communicate with a model provider directly. The gateway is the architectural basis of the sovereignty properties described in Section 12.

5.2 The two hard separations

Two asymmetries structure the runtime, corresponding to principles P4 and P5.

Fast path versus slow path. Answering the user is synchronous and consists of exactly one kind of work: filtered, fused retrieval followed by generation. Extraction, verification, deduplication, contradiction detection, consolidation, and deletion never execute inside a request. The separation mirrors the complementary-systems division discussed in Section 2.6: online behavior must be fast, and reorganization must be thorough, and the two requirements cannot be satisfied by the same process at the same time [22]. Practically, the separation is what allows the system to do more work per memory, verification, reconciliation, receipt generation, than a conventional memory layer without the user ever waiting on it.

Source of truth versus derived index. The relational database owns every fact and all metadata: memories, statuses, scopes, provenance, intervals, tasks, receipts, audit records, and the job queue itself. The vector index stores embeddings together with a payload copy of exactly the fields needed for filtered search, and nothing else. It is rebuildable at any time by a single reindex operation that streams all memories from the relational store and regenerates the index. Nothing exists only in the derived store. This asymmetry has three consequences that recur throughout the paper: disaster recovery reduces to restoring one database; embedding-model migration reduces to reindexing; and provable deletion becomes tractable, because there is exactly one authoritative record of what exists.

5.3 Transactional foundations

Every state change that must trigger downstream work, a new item ingested, a document uploaded, a deletion requested, an action approved, is recorded together with its job in a single database transaction, following the transactional outbox pattern [26]. The job queue itself is hosted in the relational database, so enqueueing is transactional by construction: it is impossible for the system to ingest an item and silently fail to process it, because the ingestion and the obligation to process it are one atomic write. Workers claim jobs with row-level locking; every job is idempotent under a deterministic key composed of the source type, the source identifier, and the job type; failures retry with exponential backoff; and jobs that exhaust their retries are parked in a dead-letter state that is visible in the governance dashboard rather than in a log file. A crashed worker loses nothing, and a duplicated delivery changes nothing.

5.4 Deployment unit

The deployment unit is one customer, one instance: an isolated composition of the two application processes, the relational database, the vector index, object storage for original files, the identity provider, and an edge proxy, brought up by a single command on EU infrastructure of the operator's choosing. Original documents are stored encrypted at rest in object storage under keys scoped by organization, user, and visibility, and retrieved only through short-lived signed URLs. Single-tenant isolation is itself a privacy mechanism: there is no shared index in which one tenant's embeddings neighbor another's, no cross-tenant cache, and no aggregate store whose access control must be trusted. The same composition, unchanged, is what a self-hosting operator runs, which makes the deployment model transparent rather than merely asserted.

5.5 Interaction surfaces: conversation and governance

The architecture surfaces to the user through two deliberately distinct interfaces, and the distinction is itself a design claim. The conversational surface is the everyday one: the user asks questions in natural language, what did I promise this client, what is open this week, prepare me for tomorrow's meeting, and the system answers from memory, drafts responses, and proposes actions that then enter the approval flow of Section 11. The conversational surface sits entirely on the fast path; every answer it produces is assembled from retrieved, source-linked facts, and every fact in an answer is presented with its status and its source, so that the answer inherits the memory model's verifiability rather than obscuring it behind fluent prose.

The governance surface, a structured dashboard, is where the memory is managed rather than used: the user can enumerate, search, inspect, edit, correct, and delete memories; see and change statuses; trace every fact to its source; review the uncertain and contradicted queues produced by verification and consolidation; browse deletion receipts in the Forgotten section; and inspect failed background jobs in the dead-letter view. The governance surface exists because of a simple observation that we regard as a principle: a user cannot correct what a user cannot see. A memory system that offers only a conversational interface makes its store effectively write-only from the user's perspective, however good its answers; correctability, the property on which the entire status vocabulary of Section 4 depends, requires enumeration, and enumeration requires a browsing surface. The two surfaces are complementary by construction: the conversation is how memory is exercised, and the dashboard is how it is governed, and neither is optional in the design.

5.6 Data and storage model

Three stores divide the system's data along lines of responsibility. The relational database holds the contractual core: the memory table, whose columns encode the model of Section 4 directly, an owner identifier and a scope enumeration, both non-nullable; a source type and source identifier, both non-nullable; the seven-state status enumeration with its default; the validity interval; the fact content and a reference to its embedding; and creation and update timestamps. Alongside it live the file metadata table, mapping each stored original to its object key, owner, scope, and sensitivity; the deletion receipt table with its hash chain; the approval and audit tables of Section 11; task and reminder records; connector synchronization state with encrypted credentials; the prompt registry with versions and measured scores; the evaluation corpus records; and the outbox and job tables of Section 5.3. That the job queue, the audit trail, and the facts share one transactional store is not incidental economy: it is what makes the atomicity guarantees of Section 5.3 and the first step of the deletion saga of Section 10 possible as single transactions.

The vector index holds, for each memory, its embedding and a payload copy of exactly the fields that retrieval preconditions require, the owner, the scope, the status, the source reference, and the validity horizon, with payload indices on the filterable fields so that the hard gates of Section 9 execute inside the index rather than after it. Object storage holds original document bytes, encrypted at rest, under keys structured as organization, then user, then scope, then a file identifier, a layout chosen so that the storage hierarchy itself mirrors the visibility model and so that keys remain stable across operational reorganizations. Access to originals is mediated exclusively by short-lived signed URLs, and the sensitivity flag gates whether a URL may be generated at all. The division of labor is summarized by principle P5: the relational store is the truth, the vector index is a rebuildable acceleration of it, and object storage is the provenance archive that makes source links literal.

6. The Ingestion Pipeline and Self-Verifying Extraction

6.1 Pipeline structure

Content enters memory only through a six-stage pipeline executed entirely on the slow path: ingest, chunk, extract, verify, embed and store, reconcile. Ingestion normalizes an item from a connector, an email, a calendar event, a note, or an uploaded document, and records it with its raw text. Chunking splits long items into pieces sized for accurate extraction; chunks are transient inputs and are never stored as memories, in accordance with principle P2. Extraction prompts a language model to produce candidate facts as structured data: the claim, the entities involved, any condition attached, and any temporal anchors, each tied to the exact source span that motivated it. Verification, described next, is the pipeline's distinctive stage. Embedding and storage write each surviving fact as a row in the relational store and a corresponding point, with its filterable payload, in the vector index. Reconciliation compares new facts against existing memory and is discussed together with consolidation in Section 7.

6.2 Verification: extraction output is a claim, not a fact

The hallucination literature summarized in Section 2.3 establishes that model output cannot be trusted by default, and that independent verification measurably improves factuality [12, 13, 14]. Cogeto internalizes this finding at the point where it matters most for a memory system: admission. Every candidate fact is submitted to a second model pass, prompted independently of the extractor, with a single question: does the cited source excerpt actually support this claim? The verifier answers supported, partial, or unsupported. Only supported facts are admitted with status active. Partial and unsupported candidates are still stored, because discarding them silently would itself be a form of opacity, but they enter with status uncertain and are queued for user review in the governance dashboard.

Three design details matter. First, the verifier's prompt is disjoint from the extractor's, so the check is not the extractor grading its own work with its own rubric; this follows the logic of Chain-of-Verification, in which verification questions are answered independently of the original draft [12]. Second, verification operates at the granularity of the atomic fact, following the observation behind FActScore that long-form output must be decomposed before it can be meaningfully verified [14]. Third, verification outcomes are recorded and become part of the published evaluation (Section 14), so the mechanism's own accuracy is itself measured. The consequence for the memory model is that the status vocabulary of Section 4 is earned: a fact marked active has passed an independent check against its source, and the user interface can say so.

6.3 Documents, originals, and extract-and-discard

Original uploaded files are retained by default, encrypted, in object storage, because provenance should be literal: a user inspecting a fact can open the exact document it came from, and the system can re-extract from originals when its pipeline improves. Retention, however, is a policy the user controls. An extract-and-discard mode, settable per upload with a per-user default, retains only the derived, verified facts and deletes the original bytes as soon as the pipeline confirms extraction. The mode exists because for some material the risk calculus favors distillation over archival, and the system should support that judgment rather than override it.

6.4 Extraction prompting and structured output

Extraction quality is determined as much by what the model is asked to produce as by the model itself, and the extraction prompt family is engineered around four commitments. First, structured output: the extractor produces machine-parseable records with fixed fields, the claim, its kind, its entities, any attached condition, and any temporal anchors, validated against a schema at the gateway boundary, so that malformed output is a handled failure rather than a stored artifact. Second, span anchoring: every candidate fact must cite the source span that motivated it, which is what gives the verification pass of Section 6.2 a well-posed question and gives the stored fact its literal provenance. Third, relative-time resolution at extraction: expressions such as next Friday or after the budget is confirmed are resolved, where resolvable, against the source's own timestamp into the validity fields of Section 4.2, because temporal anchoring performed later, without the source's context, is guesswork. Fourth, calibrated abstention: the prompt instructs, and the corpus of Section 14 tests, that a source containing nothing worth remembering yields nothing, since an extractor that always finds something is a pollution source, not a memory system. Each commitment is embodied in a versioned prompt whose measured effect on the metrics of Section 14 is part of its release record, in line with the artifact discipline of Section 13.

7. Dreaming: Autonomous Memory Consolidation

7.1 Why consolidation is necessary

A memory store that only accumulates degrades. Duplicates dilute retrieval; contradictions coexist silently; stale facts answer as if current; and the store's effective precision falls even as its recall grows. The agent-memory literature recognizes reflection and synthesis as necessary complements to accumulation [10, 11], and the biological literature is unequivocal that consolidation, the offline reorganization of recently encoded material, is constitutive of useful memory rather than an optimization of it [19, 20, 21]. Cogeto therefore treats consolidation as a first-class autonomous process with its own name, schedule, and user-facing artifact.

7.2 The nightly cycle

Dreaming runs nightly per instance, on the slow path, and comprises four passes over recent and affected memory.

Deduplication. Candidate duplicate pairs are found cheaply first, by entity overlap and embedding proximity, and confirmed by a model judgment. Confirmed duplicates are merged: one fact survives, enriched where the duplicate carried additional detail, and the other is marked replaced with a successor pointer. Merging is conservative by design; a false merge destroys a distinct fact, which is why the confirmation step exists and why merge decisions are among the evaluation harness's measured tasks.

Contradiction detection. Facts about the same entities with incompatible content are paired, and both members of the pair are marked contradicted and linked. The system does not adjudicate: choosing a winner automatically would substitute the model's judgment for the user's on precisely the questions where the user's judgment is authoritative. Instead, contradictions surface in the governance dashboard with both sources side by side, and resolution, confirming one, correcting both, or dismissing the conflict, is a user action that the audit log records.

Supersession and staleness. Where a newer fact updates an older one, the older interval is closed and the fact marked replaced, as described in Section 4.2. Facts whose validity intervals have lapsed, or whose content is time-anchored in ways that have expired, are marked outdated. Open commitments with no closing event after configurable silence are flagged for the user's attention rather than being altered.

Synthesis for the day ahead. Finally, the cycle assembles its user-facing artifact.

7.3 The dreaming digest: consolidation as a visible ritual

Background intelligence that the user never sees builds no trust. Each dreaming cycle therefore produces a morning digest of at most three lines, for example: four duplicate facts about a client were merged; a March pricing note now appears outdated in light of a newer message; two open commitments have gone quiet this week. Every line links to the exact memories, statuses, and sources involved, so the digest is not a summary of hidden work but an index into inspectable work. Nights in which nothing notable occurred produce no digest; a ritual that always fires becomes noise. The digest is where the consolidation process meets principle P1: the trust claim "the system maintains its memory responsibly" acquires a daily artifact.

7.4 On the analogy

We use the term dreaming advisedly. The analogy to sleep-dependent consolidation is architectural: like its biological counterpart, the process runs offline, reorganizes recently acquired material, integrates it with existing knowledge, extracts regularities, and resolves interference, and it exists for the same structural reason, that reorganization must not compete with real-time behavior [19, 22]. The analogy is not a claim about cognition, and nothing in the system's correctness depends on it. Its value is communicative: it names, accurately, the division of labor between a fast online path and a slow reorganizing one.

7.5 Scheduling and cost characteristics

Dreaming is scheduled, incremental, and interruptible. Each cycle operates over the day's newly admitted facts and the existing memories they touch, rather than over the whole store, so its cost grows with daily activity rather than with accumulated history; periodic full passes exist but are infrequent and resumable. Because the process runs on the slow path under the job semantics of Section 5.3, every pass is idempotent and safely repeatable: an interrupted cycle resumes without double-merging or double-marking, which is what permits the process to run on modest, shared, and even preemptible compute. The property matters architecturally: consolidation quality is bought with off-peak batch time, the cheapest resource in the system, rather than with request-time latency or dedicated capacity, which is why the design can afford to be thorough.

8. Temporal Memory and Point-in-Time Queries

The bi-temporal model of Section 4.2 supports two retrieval modes beyond the default. A point-in-time query evaluates retrieval against the validity intervals as of a specified moment: what did I believe about this engagement in March returns the facts whose intervals covered March, including facts since replaced, each labeled with its subsequent history. A change query computes the difference between two belief states: what changed on this account since the last meeting returns the facts whose status or interval changed in the window, each linked to the source that caused the change. In default retrieval, by contrast, replaced facts are excluded and outdated facts strongly demoted (Section 9); temporal queries are the explicit, user-invoked mechanism that lifts those exclusions. The combination yields what we call an undo history for machine belief: the assistant's past states are preserved, ordered, attributable, and queryable, and revising the present never falsifies the record of the past. We regard this property, more than raw recall, as what distinguishes memory from storage.

9. Retrieval: Hybrid, Fused, and Gated

9.1 Three signals, one fusion

A user question fans out to three retrieval signals with complementary strengths: semantic search over the vector index [17], which captures paraphrase and topical similarity; lexical full-text search in the relational store, in the probabilistic relevance tradition [16], which captures exact terms, identifiers, and amounts; and entity matching over the names of people, organizations, and projects, which anchors questions that are fundamentally about someone. The three ranked lists are combined by reciprocal rank fusion [15], chosen for its robustness and its freedom from score-calibration assumptions across heterogeneous retrievers.

9.2 Gates before scores

The policy layer above the fusion embodies principle P3 and is, we would argue, the retrieval design's most consequential rule: **visibility is a precondition, never a preference.** Scope and the sensitive flag are evaluated as hard filters inside each underlying query, as predicate clauses in the relational queries and as payload preconditions inside the vector search itself, so that an out-of-scope memory is never present in any candidate list at any stage. The alternative, retrieving broadly and filtering in application code, is rejected categorically: a demoted leak is still a leak, and a filter that lives in application logic is a filter that one future query will forget. Placing the gates inside the store makes the visibility rule non-bypassable by construction.

Memory quality, by contrast, is a preference, and it is expressed as post-fusion score multipliers over the status vocabulary of Section 4: active and user_approved facts carry full weight; uncertain facts are retrievable but marked; contradicted facts are retrievable with an explicit warning and both sources; outdated facts are strongly demoted; and replaced facts are excluded from default retrieval entirely, remaining reachable through the temporal queries of Section 8. The division is principled: who may see a fact is a rule, and how much a fact should be trusted is a judgment, and the architecture refuses to let the two share a mechanism.

9.3 Answer grounding

Retrieval terminates in generation, and the design constrains the hand-off. The generating model receives the fused, gated, multiplied result set together with each fact's status and source reference, and the interface renders every claim in the answer with the status and source of the facts that support it.

Facts in the uncertain or contradicted states, when they appear at all, appear visibly marked, so that the memory model's honesty about its own confidence survives into the conversational surface rather than being flattened by fluent synthesis. The commitment is modest but strict: the system never presents a demoted fact as a current one, and never presents any fact without the means to inspect it.

10. Provable Deletion

10.1 From right to mechanism

The right to erasure obliges controllers to delete personal data without undue delay on legitimate request [34]. In practice, deletion in AI memory systems is often suppression: the item stops appearing, while bytes, derived records, and index entries persist. Cogeto's position is that a deletion the user cannot verify is a deletion the user must take on faith, which contradicts principle P1. The system therefore implements deletion as a distributed protocol with a cryptographic artifact at the end.

10.2 The deletion saga

Deleting a document, or any source, spans three stores, the relational database, the vector index, and object storage, and therefore cannot be a single transaction. It is structured as a saga [25] in five steps. First, a single relational transaction removes every memory derived from the source, removes the file metadata, writes a deletion receipt row in state pending enumerating exactly what is to be removed, and enqueues the external deletions through the outbox; because provenance is mandatory (Section 4.1), the set of derived memories is computable and complete. Second, the worker deletes the corresponding points from the vector index and the original bytes from object storage, idempotently and with retries. Third, the receipt transitions to confirmed only after both external stores acknowledge. Fourth, an autonomous nightly sweep verifies that no orphaned points or objects exist for any confirmed receipt, alerting on any discrepancy; the sweep is what upgrades the receipt from a record of intent to a continuously re-verified statement of fact. Fifth, the receipt becomes a permanent, user-visible artifact.

10.3 The deletion receipt

Each receipt records the source, the counts of memories, vector points, and bytes removed, and the pending and confirmed timestamps. Receipts are signed with an instance key and hash-chained, each receipt incorporating the hash of its predecessor, following the linked time-stamping construction of Haber and Stornetta [37] and the tamper-evident audit log design of Schneier and Kelsey [38]: no receipt can later be altered or excised without breaking the chain. Receipts are permanent, exportable, and displayed in a dedicated section of the governance dashboard that we call, plainly, Forgotten. To our knowledge, a hash-chained, autonomously re-verified deletion receipt is not a feature of any widely deployed assistant memory system, and we consider it the paper's clearest instance of principle P1: the claim "I forget when told to" reduced to an artifact that a compliance officer can file.

An honest limitation accompanies the mechanism and is discussed further in Section 16: receipts prove removal from the system's stores; they cannot retroactively unsend content already processed by an external model provider under that provider's retention terms. This is precisely why the deletion protocol is paired with the isolation options of Section 12, which bound what leaves the system in the first place.

10.4 A correctness argument

The protocol's guarantees can be stated informally but precisely. Completeness of the cascade follows from the provenance invariant: because every memory carries a non-nullable reference to its source, the set of records derived from a given source is a closed, computable query, and the saga's first transaction enumerates it exactly; there is no path by which a derived fact can exist outside the enumeration, because there is no path by which a fact enters storage without provenance. Atomicity of intent follows from the outbox: the removal of the relational records, the creation of the pending receipt, and the obligation to delete externally are one transaction, so the system cannot record a deletion it will not pursue. Eventual completion follows from the job semantics: the external deletions are idempotent and retried, so transient unavailability of the vector index or object store delays, but cannot lose, the obligation. And detection of any residual divergence follows from the nightly sweep, which re-derives, from the confirmed receipts, the set of identifiers that must not exist and verifies their absence in both external stores. The receipt chain then makes the whole history tamper-evident [37, 38]. None of these properties requires trusting the operator's diligence; each is carried by a mechanism, which is the section's point.

11. Human Oversight as a State Machine

Cogeto's agents draft replies, propose tasks, summarize, and suggest actions, but consequential actions, sending a message, deleting data, writing to an external system, or bulk-modifying memory, execute only with explicit human approval. Following the human-oversight literature's emphasis on making machine initiative efficiently invocable, correctable, and dismissible [29, 30], the mechanism is a server-side state machine rather than an interface convention: a proposed action persists as a record moving through the states draft, pending_approval, approved or rejected or expired, and executed. The approval endpoint is authenticated and does exactly one thing, transition the state; execution happens only in the worker, only from the approved state, and idempotently per action. A front-end confirmation dialog, however well designed, is explicitly insufficient, because a control that exists only in the client exists only until the first alternative client. Every transition is written to the audit log, so the answer to what did the system do on my behalf, and who authorized it is a query, not an investigation.

12. The Model Layer: European by Default, Isolated by Construction

12.1 The gateway seam

Every model interaction in the system, extraction, verification, deduplication judgment, contradiction analysis, consolidation, drafting, and embedding, passes through a single module, the model gateway, whose interface is provider-neutral by requirement: it exposes completion, structured extraction, and embedding operations in shapes that presume no particular vendor. No other module in the codebase may address a model provider, and the prohibition is enforced by the same continuous-integration boundary checks that guard the other modules. The seam is a small amount of code with large consequences: it is the single point at which the system's entire relationship to model providers is defined, inspected, and changed.

12.2 Tier one: a European provider by default

The default configuration routes all model traffic to Mistral AI, a Paris-based company founded in 2023 that publishes open-weight models alongside its hosted API [39]. The choice is deliberate on three grounds. First, jurisdiction: a European provider places the system's most sensitive data flow, model inference over extracted professional content, under European law and contractual frameworks, aligned

with the sovereignty posture of the European data strategy [36]. Second, transparency culture: a provider that publishes open-weight models offers the ecosystem a degree of inspectability that pure API vendors do not. Third, continuity with the isolation tiers below: because open-weight models from the same family can be run locally, the distance between the hosted default and a fully isolated deployment is operational rather than architectural. We phrase the data-handling point carefully and verifiably: the provider offers European data residency options for API usage, and the specific contractual terms are a matter for each deployment's data processing agreement rather than for this paper.

12.3 Tier two: local pseudonymization of sensitive entities

For deployments whose users want a hosted frontier model's quality without exposing identifying details to any external party, the gateway supports a redaction tier: before any content leaves the instance, a local, CPU-based named entity recognition stage detects sensitive entities, personal names, organizations, amounts, and configurable domain-specific categories, and replaces them with consistent pseudonyms; the model's response is re-identified locally before it reaches the user or the store. The stage is built on mature open tooling: the Presidio framework for configurable PII detection and anonymization [31] and compact open-type NER models in the GLiNER family, which run efficiently on commodity CPUs [32]. Pseudonymization is a recognized safeguard in European data protection guidance [33], and its cost profile here is decisive: the stage requires roughly one gigabyte of memory, no accelerator, and milliseconds per document, so the privacy tier is effectively free at the scale of a single-tenant instance. It is worth stating what the tier is and is not: it is a strong reduction of exposure for the entity categories it covers, not a guarantee that no sensitive information can leave in free text, which is why it composes with, rather than substitutes for, the isolation tier below.

12.4 Tier three: full isolation with local models

Because the gateway is provider-neutral, an instance can be configured so that no model traffic leaves it at all. Three local configurations are supported by construction. First, local embeddings and reranking: embedding is the highest-volume model operation in the system, invoked for every stored fact and every query, and multilingual embedding models of modest size run acceptably on CPU, so localizing embeddings removes the majority of external calls at negligible cost. Second, local utility models: the background tasks of classification, deduplication confirmation, verification, and consolidation are bounded, structured judgments well within the competence of compact open-weight models served by a local runtime, and the gateway can route precisely these task classes locally while leaving user-facing generation on the hosted tier. Third, full isolation: an operator with sufficient local capacity, or an organization whose policy requires it, points the gateway's every operation at models hosted inside the deployment boundary, and the system's external model traffic becomes zero. The three configurations are settings of the same seam, not variants of the system, which is the point: sovereignty is a property of the architecture, not of a product edition. We note the honest trade-off: local utility models are selected for their task-specific adequacy, and the evaluation harness of Section 14, not vendor claims, is the arbiter of when a given task may migrate from the hosted tier to a local model without loss of measured quality.

12.6 Multilingual considerations

The system's target users work in more than one language, often within a single document, and the model layer is specified accordingly. Extraction and verification prompts are evaluated per language

against the corpus of Section 14, so that quality claims are never averages that conceal a weak language; the embedding model is selected for multilingual retrieval quality, since cross-lingual paraphrase, a commitment made in Croatian and queried in English, is a realistic retrieval case rather than an edge case; and the pseudonymization tier's entity recognition is configured per language, since names, honorifics, and organization forms are language-specific. Multilinguality also weighs on the localization path of Section 12.4: a local model adequate for English classification is not thereby adequate for the deployment's other languages, and the per-language evaluation gates are what prevent an isolation migration from silently degrading one language's quality.

12.5 Externalized memory and the forgetting problem

A final motivation for this layer's design is negative: what the system deliberately does not do. Personalization by continual fine-tuning would entangle user knowledge with model parameters, making it uninspectable, uncorrectable, and undeletable, and would expose it to catastrophic interference, the well-documented tendency of neural networks trained sequentially to overwrite prior competence [23, 24]. By externalizing all user knowledge into the non-parametric, verifiable store described in this paper, the system keeps the model stateless with respect to the user: any model, hosted or local, current or future, can serve any instance, and everything the system knows about a user remains in exactly one place, governed by exactly one set of mechanisms, including the deletion protocol of Section 10.

13. Trust Artifacts Beyond the Core

Three further mechanisms extend principle P1 from the memory core to the system's periphery.

Prompts as versioned, published artifacts. Every prompt that decides what the system remembers, the extraction, verification, deduplication, contradiction, and consolidation prompts, is versioned like a schema migration: numbered, immutable once released, and accompanied by a changelog. Because the system's core is published under an open-source license, the prompts are public, and each released version carries its measured accuracy against the evaluation corpus. The user-facing meaning is unusual and, we believe, novel in this setting: here is the exact text that decides what we remember about you, and here is how well it measurably works.

The memory passport. A user can export everything, facts, statuses, provenance links, validity history, and deletion receipts, in a single, openly documented, versioned format. The export exists as an anti-lock-in commitment: a memory system whose value compounds over time acquires a natural hold over its user, and the honest response is to make leaving easy and staying a choice.

The trust score. The evaluation results of Section 14 are published per release, in the manner of a service status page. An accuracy figure that is public, historical, and attached to specific released versions is a fundamentally different object from an internal metric: it can be checked, cited, and held against the system.

The open core. The system's source code is published under the GNU Affero General Public License, and the choice functions as a trust mechanism rather than a distribution strategy. Every enforcement claim this paper makes, that visibility gates live inside queries, that provenance is non-nullable, that the approval machine is server-side, that receipts are chained, is a claim about code, and publishing the code converts those claims from assertions into checkable statements. The license's network provision, under

which operators of modified network services must offer their modifications' source, extends the same reciprocity to the deployed form in which the system is actually consumed. Openness here is continuous with the paper's central discipline: a system whose thesis is inspectability should itself be inspectable.

14. Evaluation Methodology

14.1 The golden set

The system's quality claims rest on a hand-labeled corpus, the golden set, of realistic items, notes, emails, calendar events, and document excerpts, each annotated with the facts a diligent human assistant would extract, the entities and conditions involved, expected temporal anchors, expected verification outcomes, and, for designed pairs, expected deduplication and contradiction relations. The corpus begins at fifty to one hundred items per supported language, includes deliberately hard cases, conditional commitments, relative dates, multi-fact sources, sources containing nothing worth remembering, and near-duplicate and contradicting pairs, and grows permanently: every interesting real-world failure becomes a new case, reconstructed synthetically so that the corpus itself contains no real user data. Matching between extracted and expected facts is semantic rather than literal, under a fixed and versioned similarity threshold, so that scores remain comparable across releases.

14.2 Metrics and gates

Five measurements are computed on every change to prompts, models, or pipeline code: extraction precision and recall against the labeled facts; verification agreement with the labeled verification outcomes; deduplication accuracy over the designed pairs, with false merges penalized more heavily than missed merges, since a false merge destroys a distinct fact; and contradiction detection precision and recall. Each metric carries a threshold that operates as a continuous-integration gate: a change that regresses a metric below its gate does not ship, and thresholds may be raised but never lowered without a recorded decision. The methodological commitment is that the pipeline's most failure-prone components, the model-mediated judgments, are the most continuously measured ones, and that the measurement is built alongside the extractor rather than after it, because an unmeasured extraction pipeline is an unfalsifiable one.

The initial gates are stated explicitly, both as engineering targets and as the reference against which the published scores of Section 14.3 will be read:

Metric	Initial gate	Rationale
Extraction precision	at least 0.85	A stored falsehood is durable; precision is weighted over recall at admission
Extraction recall	at least 0.80	Missed commitments are the product failure the user notices most
Verification agreement	at least 0.90	The admission check must itself be dependable before its verdicts gate status
Deduplication accuracy	at least 0.90	False merges destroy distinct facts and are penalized above missed merges
Contradiction recall	at least 0.70	Conservative detection with user adjudication tolerates misses better than false alarms

Gates may be raised as the system matures; lowering a gate requires a recorded, published decision, so that the trajectory of the thresholds is itself part of the auditable record.

14.3 Publication

The same figures, per release, per language, with corpus sizes and harness versions, constitute the published trust score of Section 13. We are aware that publishing one's error rates is unusual; we consider it the evaluation methodology's point. A system whose thesis is that trust claims require artifacts cannot exempt its own accuracy claims.

15. Deployment, Sovereignty, and Regulatory Alignment

The deployment model described in Section 5.4, one isolated instance per customer on EU infrastructure, composes with the mechanisms of the preceding sections into a specific regulatory posture. Data protection by design and by default [34, Art. 25] is addressed structurally: single-tenant isolation, mandatory provenance, hard visibility gates, encryption at rest, pseudonymization options, and minimal external data flow are properties of the architecture rather than configurations of it. The right to erasure [34, Art. 17] is addressed by the deletion protocol and its receipts, which give the data subject not only the deletion but the evidence of it. Transparency expectations of the kind the Artificial Intelligence Act introduces [35] are addressed by the system's disposition to show its work: source links on every fact, statuses on every belief, audit records on every action, published prompts, and published accuracy. And the broader European ambition of data sovereignty [36] is addressed at the only level where it is meaningful, the level of where bytes reside and where inference runs: on infrastructure the operator chooses, with a European provider by default and full isolation available by configuration. We do not claim that architecture alone constitutes compliance, which is always also organizational and contractual; we claim that this architecture makes the technical side of compliance demonstrable rather than asserted.

15.1 Threat model and security posture

The security design is organized around four adversary classes. Against an external network adversary, the instance exposes a single hardened edge: transport security terminates at the proxy, all application endpoints require authentication through the identity seam, and original documents are reachable only through short-lived signed URLs whose issuance the sensitivity flag gates. Against a curious or compromised co-tenant, the answer is structural: there is no co-tenant, since isolation is per customer at the instance level, and no shared index, cache, or queue exists in which one tenant's data could neighbor another's. Against an adversary with read access to storage media, data at rest is encrypted in object storage and on the host's volumes, and connector credentials are additionally encrypted at the application layer before storage. Against the most subtle class, the system's own future operators and code, the defenses are the ones this paper has described as trust mechanisms: the audit log is append-only and its deletion receipts are hash-chained, so tampering with the record of what was done, approved, or deleted is detectable [37, 38]; consequential actions execute only through the approval state machine; and secrets never enter the repository or the images, arriving only as per-instance configuration. Logging is subject to a strict redaction rule: memory content, document text, and credentials never appear in logs, so that observability does not become a shadow copy of the data the rest of the architecture guards. We state the residual risks plainly: a compromised host compromises its instance, as in any single-tenant design, and the blast radius of that event is one customer, which is the isolation model's purpose; and the external

model tier of Section 12.2 is governed by contract and by the minimization tiers of Sections 12.3 and 12.4 rather than by the instance's own controls.

15.2 Operational reliability

The reliability posture follows from principle P5 and the job semantics of Section 5.3. Backup is one problem, not three: the relational store, holding all facts, metadata, receipts, and queue state, is dumped nightly to storage in a second location; original documents are replicated at the object-storage layer; and the vector index is deliberately not backed up at all, because the reindex operation reconstructs it from the source of truth, which turns the most annoying store to back up into the easiest one to recover. Restore, correspondingly, is a rehearsed procedure rather than an assumption. Failure handling is uniform: every background job is idempotent and retried with backoff; jobs that exhaust retries are parked in the dead-letter state and surfaced in the governance dashboard, so that failure is a visible object the user or operator can inspect and re-run rather than a log line; and the nightly deletion sweep of Section 10.2 doubles as a continuous integrity check on the system's most sensitive invariant. Upgrades across a fleet of single-tenant instances use one versioned artifact per release, database migrations applied by a one-shot initialization step before the application starts, canary deployment to internal instances first, and rollback by artifact version plus, where the embedding model changed, reindex. The design's operational thesis is the same as its trust thesis: fewer moving parts, one authority for state, and every failure mode terminating in an inspectable artifact rather than a silent divergence.

16. Discussion

16.1 An end-to-end illustration

A compressed scenario ties the mechanisms together. On Monday evening a consultant forwards a client email into her instance and jots a note: send the revised proposal after the client confirms the budget. Ingestion records both items and, in the same transactions, enqueues their processing. Overnight, on the slow path, extraction produces candidate facts, among them a commitment with a person, a deliverable, and a condition; the verification pass confirms each against its source span, and the supported facts are stored as active, embedded, and indexed, with the conditional commitment carrying an open validity interval and its condition. The dreaming cycle, running later that night, notices that the new commitment duplicates an older, vaguer note, merges them with the older fact marked replaced, and detects that a price mentioned in the email contradicts a figure recorded in March; both price facts are marked contradicted and linked. Tuesday morning the consultant's digest reads: merged two notes about the proposal; a March price now conflicts with yesterday's email. She taps the second line, sees both sources side by side, confirms the new figure, and the March fact's interval closes as she does so, its history preserved. Before her ten o'clock she asks the conversational surface to prepare her for the meeting; the fast path retrieves, under her scope gate, the active facts about the client, presents the open commitment with its condition and source, and drafts a follow-up message, which enters the approval queue and is sent only after her explicit confirmation, the transition audit-logged. Months later the engagement ends and the client requests erasure; she deletes the engagement's documents, the saga removes the originals, every derived memory, and every vector, and the Forgotten section receives a confirmed, signed receipt she forwards to the client. Every step in the scenario is a mechanism from Sections 4 through 12, and every claim in it corresponds to something on her screen.

16.2 Why the design works

The paper's mechanisms are individually modest: an enum, a NOT NULL constraint, a second model call, a nightly batch job, a saga, a hash chain, a fused query, a state machine. None is exotic, and that is deliberate. The design's force comes from three disciplined combinations. First, the closed memory model converts quality goals into checkable invariants, so that the system's correctness claims are located in schema and state machines rather than in prose (Section 4.3). Second, the fast path and slow path separation buys the time to be careful: verification, reconciliation, consolidation, and receipt generation are affordable precisely because they never block a response, which is how the system can do strictly more diligence per memory than an accumulator without feeling slower. Third, the artifact rule closes the loop between mechanism and trust: each mechanism terminates in something a user, an auditor, or a partner can inspect, a receipt, a diff, a score, a source, a log, an export, so the system's trustworthiness is a property the user verifies rather than a property the operator asserts.

16.3 What is new

We claim novelty not for any component but for the assembly and for three specific commitments within it. The hash-chained, autonomously re-verified deletion receipt gives the right to erasure an operational artifact we have not encountered in deployed assistant memory systems. The admission-time independent verification pass, with its outcomes feeding both the status vocabulary and a published score, moves hallucination mitigation from generation time, where the literature has concentrated, to memory formation time, where an error becomes durable. And the publication of versioned prompts with measured accuracy extends open-source transparency from code to the model-facing layer that actually determines the system's behavior.

16.4 Limitations

Several limitations deserve explicit statement. The system is under development, and the quality thresholds of Section 14 are commitments to be met, not results being reported; this paper is a design argument, not an evaluation. The verification pass reduces but cannot eliminate extraction error, and its own judgments are model-mediated; the mitigation is measurement, not certainty. The pseudonymization tier covers configured entity categories and cannot guarantee that no sensitive information leaves in free text; the mitigation is the isolation tier. Deletion receipts prove removal from the system's stores and cannot retroactively affect content already processed externally under a provider's terms; the mitigation is minimizing and bounding what is sent externally in the first place. Contradiction detection is conservative and deliberately non-adjudicating, which trades autonomy for correctness and places a review burden on the user. And the single-tenant model, while central to the privacy posture, multiplies operational cost per user relative to pooled architectures; we regard the trade as the price of the isolation claims and have shaped the architecture, one artifact, two processes, few services, to keep that price low.

16.5 Future work

Beyond completing the implementation and reporting evaluation results against the published gates, three directions follow naturally from the design. First, richer temporal reasoning: the bi-temporal store supports point-in-time and change queries today, and we intend to investigate periodicity, expectation, and trend extraction over the same substrate. Second, local-model migration: as compact open-weight models improve, the evaluation harness provides a principled, measured path for moving successive task classes from the hosted tier to full isolation, and we intend to publish those measurements. Third,

interoperability of the memory passport format, so that verifiable memory can become a property users carry rather than a property of any single system.

17. Conclusion

We have proposed Cogeto, a memory architecture for LLM assistants organized around a single discipline: every claim the system makes about its own trustworthiness must terminate in an artifact that someone outside the system can inspect. The discipline yields a closed, bi-temporal, provenance-mandatory memory model; an admission pipeline in which facts are verified before they are believed; a nightly consolidation process that keeps the store honest and shows its work each morning; retrieval in which visibility is physics rather than policy; deletion that ends in a signed, chained, re-verified receipt; oversight that is a state machine rather than a dialog box; and a model layer that is European by default and fully isolable by construction. Individually, these are established techniques applied with care. Together, they constitute a proposal about what machine memory must become as assistants take custody of professional life: not larger, but accountable. Most systems in this space remember. The system described here is being built to remember responsibly, and to prove it.

Acknowledgments

The author thanks the collaborators of MVT Solutions Group d.o.o. and MCTO Advisory d.o.o. for discussions that shaped this work. Correspondence: ivan@themrcto.com; project information: cogeto.eu.

Appendix A. Terminology

The following closed vocabulary is used consistently throughout the system's code, schema, interfaces, and documentation; this paper uses it in the same senses.

Term	Definition
Memory	One stored, extracted, verified fact carrying owner, scope, provenance, status, and validity interval; the system's atomic unit (Section 4.1)
Source / provenance	The exact originating item, message, event, note, document, or user statement, that every memory non-nullably references (Section 4.1)
Scope	The visibility class of a memory, private or shared, enforced as a hard precondition in every read path (Sections 4.1, 9.2)
Status	The memory quality state, one of seven: active, user_approved, uncertain, contradicted, outdated, replaced, sensitive (Section 4.1)
Validity interval	The valid-time span during which a fact holds, maintained independently of its ingestion time (Section 4.2)
Supersession	The closing of an older fact's interval and its replacement by a successor, with history preserved (Section 4.2)
Ingestion pipeline	The six-stage slow-path admission process: ingest, chunk, extract, verify, embed and store, reconcile (Section 6.1)
Verification pass	The independent model check that a candidate fact's cited source supports it, gating admission to the active state (Section 6.2)
Dreaming	The nightly consolidation cycle performing deduplication, contradiction detection, supersession, and staleness marking (Section 7)
Dreaming digest	The at-most-three-line morning artifact indexing the night's consolidation work, each line linked to its objects (Section 7.3)
Fast path / slow path	The synchronous retrieval-and-answer path versus the asynchronous maintenance path; the former never performs the latter's work (Section 5.2)
Hard gate	A visibility or sensitivity precondition evaluated inside the store's queries, never expressible as a score adjustment (Section 9.2)
Status multiplier	The post-fusion score factor applied per quality state in default retrieval (Section 9.2)
Deletion saga	The five-step compensating protocol that removes a source, its derived memories, and their vectors across all stores (Section 10.2)
Deletion receipt	The signed, hash-chained, autonomously re-verified record of a completed deletion, permanent and exportable (Section 10.3)
Approval state machine	The server-side lifecycle, draft, pending approval, approved or rejected or expired, executed, through which every consequential action passes (Section 11)
Model gateway	The single module through which all model and embedding calls pass; the seam at which providers are configured, localized, or fully isolated (Section 12.1)
Redaction tier	The local, CPU-based pseudonymization of sensitive entities before any external model call (Section 12.3)
Golden set	The hand-labeled, per-language evaluation corpus against which extraction, verification, deduplication, and contradiction detection are continuously measured (Section 14.1)
Trust score	The published, per-release evaluation results, presented with corpus sizes and harness versions (Sections 13, 14.3)

Term	Definition
Memory passport	The complete, openly documented export of a user's facts, statuses, provenance, history, and receipts (Section 13)
Governance dashboard	The structured surface for enumerating, inspecting, correcting, and deleting memory, and for reviewing queues, receipts, and failed jobs (Section 5.5)

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems 30 (NeurIPS 2017), pp. 5998-6008. arXiv:1706.03762.

[2] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pp. 1877-1901. arXiv:2005.14165.

[3] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuettler, H., Lewis, M., Yih, W., Rocktaeschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pp. 9459-9474. arXiv:2005.11401.

[4] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of EMNLP 2020, pp. 6769-6781. doi:10.18653/v1/2020.emnlp-main.550.

[5] Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. (2020). REALM: Retrieval-Augmented Language Model Pre-Training. Proceedings of the 37th International Conference on Machine Learning (ICML 2020), PMLR vol. 119. arXiv:2002.08909.

[6] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., and Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997.

[7] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of Hallucination in Natural Language Generation. ACM Computing Surveys 55(12), Article 248, pp. 1-38. doi:10.1145/3571730.

[8] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., and Liu, T. (2025). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. ACM Transactions on Information Systems 43(2), pp. 1-55. doi:10.1145/3703155. arXiv:2311.05232.

[9] Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv preprint arXiv:2310.08560.

[10] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST 2023), Article 2, pp. 1-22. doi:10.1145/3586183.3606763.

[11] Zhang, Z., Bo, X., Ma, C., Li, R., Chen, X., Dai, Q., Zhu, J., Dong, Z., and Wen, J.-R. (2024). A Survey on the Memory Mechanism of Large Language Model based Agents. arXiv preprint arXiv:2404.13501.

[12] Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., and Weston, J. (2024). Chain-of-Verification Reduces Hallucination in Large Language Models. Findings of the Association for Computational Linguistics: ACL 2024, pp. 3563-3578. arXiv:2309.11495.

[13] Manakul, P., Liusie, A., and Gales, M. J. F. (2023). SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. Proceedings of EMNLP 2023, pp. 9004-9017. doi:10.18653/v1/2023.emnlp-main.557.

[14] Min, S., Krishna, K., Lyu, X., Lewis, M., Yih, W., Koh, P. W., Iyyer, M., Zettlemoyer, L., and Hajishirzi, H. (2023). FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. Proceedings of EMNLP 2023, pp. 12076-12100. doi:10.18653/v1/2023.emnlp-main.741.

[15] Cormack, G. V., Clarke, C. L. A., and Buettcher, S. (2009). Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009), pp. 758-759. doi:10.1145/1571941.1572114.

[16] Robertson, S., and Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval 3(4), pp. 333-389. doi:10.1561/1500000019.

[17] Malkov, Yu. A., and Yashunin, D. A. (2020). Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 42(4), pp. 824-836. doi:10.1109/TPAMI.2018.2889473. arXiv:1603.09320.

[18] Kulkarni, K., and Michels, J.-E. (2012). Temporal Features in SQL:2011. ACM SIGMOD Record 41(3), pp. 34-43. doi:10.1145/2380776.2380786.

[19] Diekelmann, S., and Born, J. (2010). The Memory Function of Sleep. Nature Reviews Neuroscience 11(2), pp. 114-126. doi:10.1038/nrn2762.

[20] Stickgold, R. (2005). Sleep-dependent Memory Consolidation. Nature 437(7063), pp. 1272-1278. doi:10.1038/nature04286.

[21] Rasch, B., and Born, J. (2013). About Sleep's Role in Memory. Physiological Reviews 93(2), pp. 681-766. doi:10.1152/physrev.00032.2012.

[22] McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why There Are Complementary Learning Systems in the Hippocampus and Neocortex: Insights from the Successes and Failures of Connectionist Models of Learning and Memory. Psychological Review 102(3), pp. 419-457. doi:10.1037/0033-295X.102.3.419.

[23] French, R. M. (1999). Catastrophic Forgetting in Connectionist Networks. Trends in Cognitive Sciences 3(4), pp. 128-135. doi:10.1016/S1364-6613(99)01294-2.

[24] McCloskey, M., and Cohen, N. J. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In Bower, G. H. (ed.), The Psychology of Learning and Motivation, Vol. 24, pp. 109-165. Academic Press.

[25] Garcia-Molina, H., and Salem, K. (1987). Sagas. Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, pp. 249-259. doi:10.1145/38713.38742.

[26] Richardson, C. (2018). Microservices Patterns: With Examples in Java. Manning Publications, Shelter Island, NY. ISBN 978-1-61729-454-9.

[27] Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional, Boston. ISBN 978-0-321-12521-5.

[28] Fowler, M. (2015). MonolithFirst. martinfowler.com, 3 June 2015. https://martinfowler.com/bliki/MonolithFirst.html.

[29] Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., and Horvitz, E. (2019). Guidelines for Human-AI Interaction. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1-13. doi:10.1145/3290605.3300233.

[30] Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., and He, L. (2022). A Survey of Human-in-the-loop for Machine Learning. Future Generation Computer Systems 135, pp. 364-381. doi:10.1016/j.future.2022.05.014.

[31] Microsoft Presidio: Data Protection and De-identification SDK. Open-source software and documentation. https://microsoft.github.io/presidio.

[32] Zaratiana, U., Tomeh, N., Holat, P., and Charnois, T. (2024). GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer. Proceedings of the 2024 Conference of the North American Chapter of the ACL: Human Language Technologies (Volume 1: Long Papers), pp. 5364-5376. doi:10.18653/v1/2024.naacl-long.300.

[33] ENISA, European Union Agency for Cybersecurity (2019). Pseudonymisation Techniques and Best Practices: Recommendations on Shaping Technology According to Data Protection and Privacy Provisions. November 2019.

[34] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data (General Data Protection Regulation). Official Journal of the European Union, L 119, 4 May 2016, pp. 1-88.

[35] Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union, OJ L, 2024/1689, 12 July 2024.

[36] European Commission (2020). A European Strategy for Data. Communication COM(2020) 66 final, Brussels, 19 February 2020.

[37] Haber, S., and Stornetta, W. S. (1991). How to Time-Stamp a Digital Document. Journal of Cryptology 3(2), pp. 99-111. doi:10.1007/BF00196791.

[38] Schneier, B., and Kelsey, J. (1999). Secure Audit Logs to Support Computer Forensics. ACM Transactions on Information and System Security 2(2), pp. 159-176. doi:10.1145/317087.317089.

[39] Mistral AI (2025). About Mistral AI. Company information. https://mistral.ai/about.

[40] Stanford Institute for Human-Centered Artificial Intelligence (2025). The 2025 AI Index Report, 8th edition, Economy chapter. Stanford University. https://hai.stanford.edu/ai-index/2025-ai-index-report.

[41] Cisco (2024). 2024 Data Privacy Benchmark Study. Cisco Systems, San Jose, January 2024.

[42] DLA Piper (2025). GDPR Fines and Data Breach Survey: January 2025, 7th annual edition. DLA Piper LLP.