# Smriti– Production-Realistic Scope

This is a senior-engineer-level V1. It is not a demo, not a college project, and not enterprise fantasy. The goal of V1 is **trust, correctness, and control**, not breadth.

---

## V1 Product Goal (Non-Negotiable)

Build a **reliable, long-term memory engine for LLM-based systems** that:

- Learns over time
- Does not corrupt itself
- Controls token cost
- Handles contradictions explicitly
- Can be safely used in production

If V1 does not achieve this, nothing else matters.

---

## What V1 IS (Clear Scope)

### 1. Text-Only Memory (Deliberate choice)

V1 handles **text and structured metadata only**.

- Conversations
- Documents
- Tool outputs
- System observations

Multimodal is explicitly **out of scope** for V1.

Reason: multimodal pipelines add failure modes without improving core intelligence.

---

### 2. Explicit Memory Types (Required)

#### a) Working Memory

- Current conversation state
- Active task context
- Hard token budget enforcement
- Auto-summarization with traceability

Purpose: prevent context explosion.

---

#### b) Episodic Memory

- Time-ordered interactions
- Task execution logs
- Conversation summaries
- Immutable by default

Purpose: historical truth.

---

#### c) Semantic Memory

- Extracted facts
- User preferences
- Learned rules
- System beliefs

Purpose: long-term knowledge.

---

### 3. Memory Lifecycle (Core of V1)

Every memory object MUST support:

- Create
- Retrieve
- Update (versioned)
- Decay (soft delete)
- Archive
- Hard delete

No exceptions.

Memory lifecycle is rule-driven first, LLM-assisted second.

---

## Contradiction Handling (Mandatory in V1)

V1 must never silently overwrite memory.

### Required behavior:

- Conflicting memories are both stored
- One becomes **active**, others **inactive**
- Resolution rules:
- Recency

- Source trust score
- Confidence score
- User confirmation (optional)

Semantic memory is **versioned belief**, not absolute truth.

---

## Memory Storage (Minimal but Real)

### Required components:

- Vector DB (Qdrant or equivalent)
- Relational DB (Postgres or SQLite)
- In-process cache (Redis optional)

Graph DB is **explicitly deferred**.

---

## Retrieval Strategy (Not Naive RAG)

Retrieval must:

- Be memory-type aware
- Respect time windows
- Enforce relevance thresholds
- Respect active/inactive states

No “dump everything into prompt”.

---

## LLM Usage Rules (Hard Constraints)

LLMs MAY:

- Summarize
- Classify
- Suggest memory updates

LLMs MAY NOT:

- Delete memory
- Resolve contradictions alone
- Write semantic facts without confidence

Humans and rules always win.

---

## APIs (V1 Only)

**Core APIs:**

- store_memory()
- retrieve_memory()
- update_memory()
- decay_memory()
- resolve_conflict()

No plugins. No magic.

---

## Observability (Required for Trust)

Every memory operation must log:

- Why it was stored
- Who/what triggered it
- Confidence score
- Previous versions

Debuggability is a feature.

---

## Security & Privacy (Minimal but Real)

V1 includes:

- Encryption at rest
- Namespaced memory scopes
- User-initiated delete

Compliance certifications are **out of scope**.

---

## What V1 EXPLICITLY EXCLUDES

- Audio / video / vision
- Emotion detection
- Auto-agent swarms
- Enterprise RBAC
- Kubernetes-scale infra
- AGI claims

These are postponed, not denied.

## Success Criteria (Objective)

V1 is successful if:

- Memory remains consistent after weeks of use
- Token usage decreases over time
- Contradictions are explainable
- A developer trusts it enough to keep it on

## V1 Deliverables

- Core memory engine (Python)
- Local-first deployment
- Clear documentation
- Reproducible examples
- Benchmarks vs naive RAG

## Why This V1 Will Actually Work

Because it:

- Solves one hard problem deeply
- Avoids premature scale
- Prioritizes correctness over features
- Builds trust before ambition

## Engineer’s Final Note

If you build this V1 cleanly, V2 becomes obvious. If you skip this discipline, no amount of features will save the project.

This is the foundation. Everything else is optional.