
🧾 Day 3 Whitepaper Notes

Context Engineering — Sessions & Memory

🧭 1. Purpose & Core Idea

This paper focuses on how agents **retain, recall, and reuse knowledge** across interactions — the backbone of *continuity and adaptability* in agentic systems.

> Agents don't become intelligent by remembering everything; they become intelligent by remembering *the right things at the right time*.

Context Engineering is the discipline of deciding **what information matters**, **when to forget**, and **how to recall efficiently**.

🧩 2. The Anatomy of Context

An agent's context is a *living state* built from multiple layers — each with distinct purpose, scope, and lifetime.

Layer	Scope	Examples
Session Context	Active conversation state (unique ID, metadata, intent)	"User is configuring sales report filters."
Short-Term Memory (STM)	Recent steps within a session	Last 10 turns or API calls
Long-Term Memory (LTM)	Persistent knowledge across sessions	User preferences, past decisions, FAQs

Each layer is bounded by *cost, relevance,* and *privacy constraints*.

🧠 3. Session Model and Lifecycle

Sessions act as the unit of continuity between the user and agent.

Lifecycle:

1. 🆕 *Create* — new session ID & metadata (user, goal, timestamp)
2. 💬 *Active* — ongoing reason–act–observe loop
3. ⏸️ *Suspended* — temporarily paused but retained in memory
4. ▶️ *Resumed* — context restored and continued
5. 📦 *Archived* — stored for retrieval or long-term analytics

Sessions enable multi-turn reasoning with traceability, similar to how a thread in Slack or an HTTP session retains state.

🧱 4. Memory Architectures Explained

a. Buffer Memory

Stores recent N interactions (verbatim). Used for short, bounded contexts.

b. Summary Memory

Compresses older turns into semantic summaries to prevent token overload while preserving intent.

c. Vector Memory

Encodes past events into embeddings for semantic retrieval (using cosine similarity or ANN).

d. Hybrid Memory

Combines summary + vector for optimal accuracy and cost control.

Core principle: *The agent should never re-read everything — it should recall what matters.*

🔁 5. Context Management Loop

Each agent iteration follows this cycle:

1. **Retrieve Context** → collect session state + relevant memory entries.
2. **Generate Reasoning** → produce next action or tool call.
3. **Execute & Observe** → record outcomes.
4. **Update Memory** → append or summarize new facts.
5. **Trim Context** → remove stale or irrelevant details.

This loop keeps the agent's attention window focused and its reasoning grounded in past reality.

🧩 6. Key Patterns in Session & Memory Design

Pattern	Purpose	Implementation Hint
Session Token Chaining	Maintain state across API calls	Use UUID per conversation thread
Checkpointing	Save intermediate reasoning steps	Periodic serialization
Summarization Loop	Prevent context overflow	Summarize every N turns
Hierarchical Memory	Speed retrieval	Split by topic or intent
Memory Expiry Policies	Data retention compliance	TTL for each memory type

---

⚙️ 7. Engineering Challenges & Best Practices

- **Context Overflow → Trim Early:** Use summary and embedding indexes to control token growth.
- **State Isolation:** Separate user context from system context to avoid leaks across agents.
- **Memory Security:** Encrypt stored sessions & mask sensitive data (PII).

- **Cross-Session Learning:** Sync long-term memory for multi-agent collaboration.
- **Observability:** Each session should be traceable via logs, spans, and telemetry IDs for debugging.

🧱 8. Implementation in Google ADK (Notebooks Summary)

🧩 Notebook 1: *Agent Sessions*

- Implemented a `SessionManager` class to create and track session objects.
- Added metadata (creation time, user ID, goal, status).
- Used ADK to resume/suspend sessions programmatically.
- Observed context continuity after pause/resume — agent picked up mid-conversation without losing intent.

🧠 Notebook 2: *Agent Memory*

- Built a `MemoryProvider` class for short-term, long-term, and hybrid storage.
- Integrated vector store retrieval for semantic recall.
- Implemented context summarization pipelines using LLM compression.
- Tested “reflective memory” — agent summarizes its own experience to learn patterns for next sessions.

🔒 9. Privacy & Ethical Considerations

The paper warns that memory can become a “shadow database.” Key safeguards:

- Explicit user consent for long-term storage.

- Data minimization & retention limits.
- Differential privacy for shared context.
- Human-in-the-loop review for sensitive recall events.

🧬 10. Real-World Applications

Use Case	Example
Personalized Assistants	Remember user preferences and projects across weeks.
Enterprise Agents	Maintain task state for tickets or sales analysis.
Collaborative Systems	Share context between Planner and Executor agents.
Analytics Memory Agents	Summarize daily insights from SQL queries.

💡 11. Core Takeaways

1. **Context = Intelligence.** A model without memory is a chatbot; with context, it becomes a colleague.
2. **Sessions are stateful threads** for continuity.
3. **Memory is not storage — it's strategy.**
4. **Summarization and retrieval** together enable scalable agentic reasoning.
5. **Security and Observability** must be baked into memory design.