SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 04 · Advanced Agentic Patterns

17

Agent Memory & Persistence

Long-term persistent and structured episodic memory stores

Module 17 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Long-term persistent and structured episodic memory stores. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	4	17/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 04

02 Learning Objectives

- Differentiate short-term, episodic, and semantic long-term memory.
- Persist and retrieve memories across sessions.
- Add a managed memory layer (Mem0 / Zep).
- Prevent context bloat with summarisation and retrieval.

03 Core Concepts

Memory taxonomy

Short-term lives in the context window. Episodic memory stores past events/interactions for recall. Semantic memory holds distilled facts and preferences. Procedural memory holds learned how-to. Production agents need durable episodic + semantic stores to feel coherent over time.

Persistence patterns

Write important interactions to a store (vector + structured), then retrieve the most relevant at the start of each turn. This keeps continuity without stuffing the whole history into context, which is costly and degrades attention.

Managed memory layers

Mem0 and Zep provide extraction, storage, and retrieval of user/agent memory with decay and relevance ranking. They save you from rebuilding memory plumbing and add features like fact deduplication and temporal awareness.

Context hygiene

Summarise long histories, retrieve selectively, and expire stale memories. Treat the context window as a scarce cache, not a transcript — quality of what you load beats quantity.

04 Visual Diagram

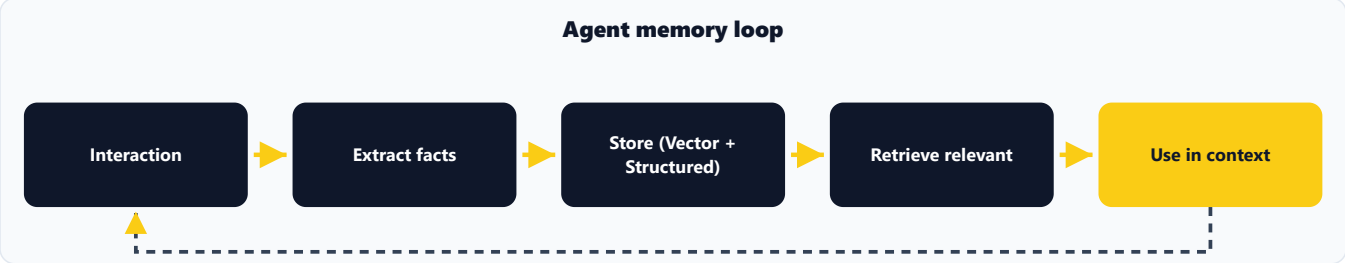

Figure 17 — Write important facts to a durable store, then retrieve only the relevant ones each turn.

05 Key Frameworks & Tools

Mem0 | Zep | Vector DB | Summarisation chains

06 Hands-On Build

🛠 Mem0 + Zep persistent memory layer

1. Add a memory store and write key facts after each interaction.
2. Retrieve top relevant memories at the start of each turn.
3. Integrate Mem0/Zep for managed extraction and decay.
4. Add summarisation to cap context size on long sessions.

07 Code Walkthrough

Persisting and recalling with Mem0 — PYTHON

```python
from mem0 import Memory

m = Memory()
m.add("User prefers concise answers and uses Python.", user_id="u1")

relevant = m.search("How should I format replies?", user_id="u1")
for r in relevant:
    print(r["memory"])
```

08 🧪 Lab Assistant

Lab 17 · Persistent memory with Mem0

Prerequisites:pip install mem0ai · An API key

1. **Add a user preference to memory**
   > Expected: The fact is stored under the user id
2. **Search memory before a new turn**
   > Expected: Relevant memories returned, ranked
3. **Start a fresh session and recall**
   > Expected: Continuity across sessions

Troubleshooting

- Stale facts surface → add decay/expiry.
- Context bloats → retrieve top-k, don't dump history.

09 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Dumping entire chat history into context instead of retrieving.
- Never expiring memories, so stale facts mislead the agent.
- Storing secrets/PII in memory without controls.

✓ KEY TAKEAWAYS

- Coherent agents need durable episodic + semantic memory.
- Retrieve selectively; the context window is a scarce cache.
- Mem0/Zep provide managed extraction, decay, and recall.

📺 **Companion video:** Watch the module 17 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

10 Further Resources

Mem0 docs • Zep docs • Cognitive architecture papers