

# Reverse Engineering Latest ChatGPT

## Memory Feature (And Building Your Own)

Prasad Thammineni

April 23, 2025 • 10 min read

In this analysis, we break down OpenAI's new memory system and provide a technical blueprint for implementing similar capabilities, without requiring programming expertise.

### New to AI Memory Systems?

If you're new to the concept of memory in AI agents, we recommend first reading our foundational article [Beyond Stateless: How Memory Makes AI Agents Truly Intelligent](#), which explains the fundamental concepts and importance of memory in AI systems.

### OpenAI's Memory Evolution

Last week, OpenAI announced a significant upgrade to ChatGPT's memory capabilities. The enhanced system now allows ChatGPT to reference a user's entire conversation history across multiple sessions—transforming it from a stateless responder into a more personalized assistant that evolves with ongoing interactions.

This update represents a major advancement in how commercial AI systems handle persistent user context, pointing toward what Sam Altman described as "AI systems that get to know you over your life, and become extremely useful and personalized."

### Key Components of OpenAI's Memory Architecture

Let's break down the likely architecture behind OpenAI's implementation:

System Architecture

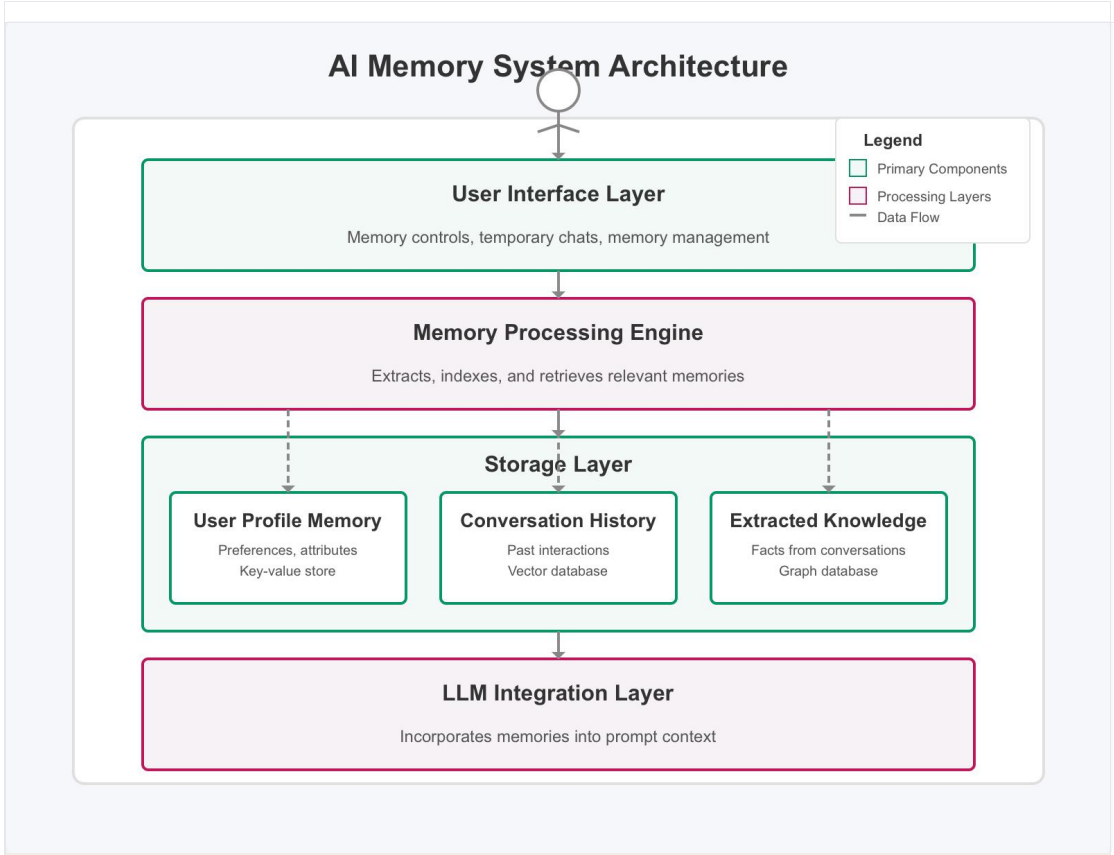
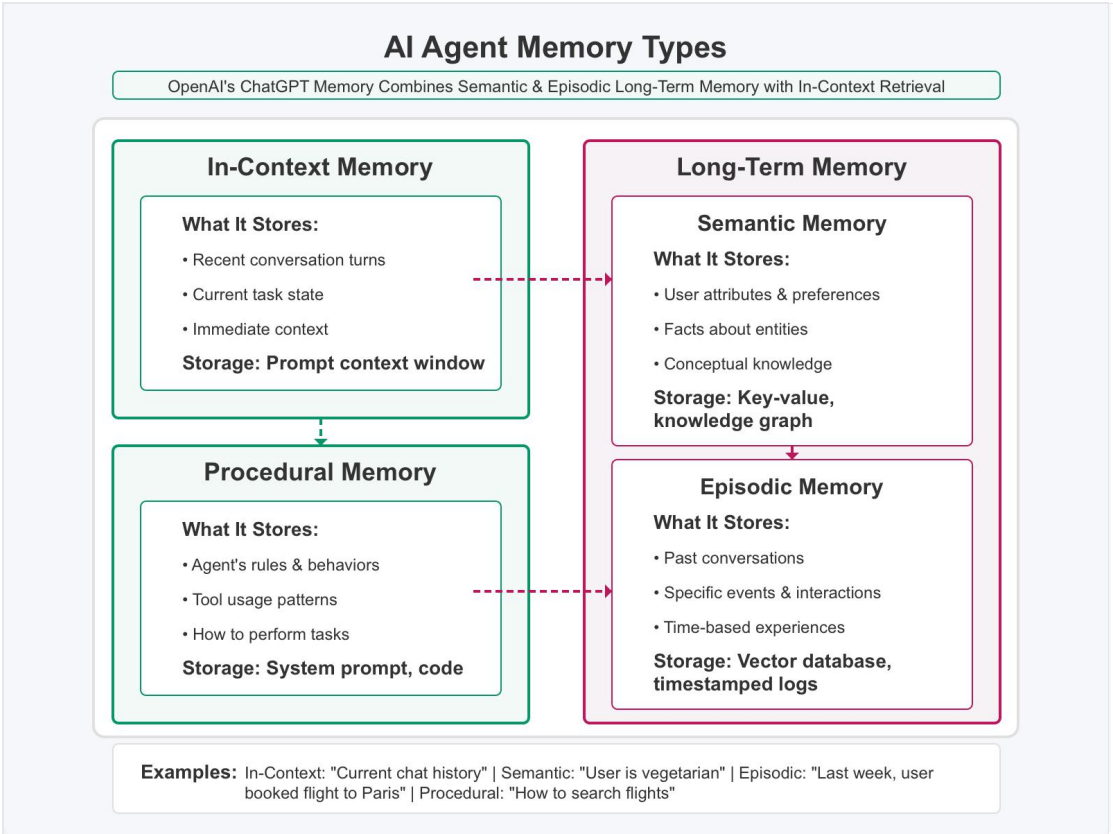


Figure 1: Memory System Architecture

The architecture consists of four primary components working together as a cohesive system. At the top level, the User Interface Layer provides controls for memory visibility, temporary chats, and memory management—giving users transparency and control over what’s remembered. Below that, the Memory Processing Engine works continuously to extract, index, and retrieve relevant memories from conversations. This connects to the Storage Layer, which maintains different types of memory with varying levels of persistence. Finally, the LLM Integration Layer incorporates these memories into the prompt context, ensuring the AI has access to the right information at the right time.

Memory Types and Storage

OpenAI’s system likely implements multiple types of memory working in concert:

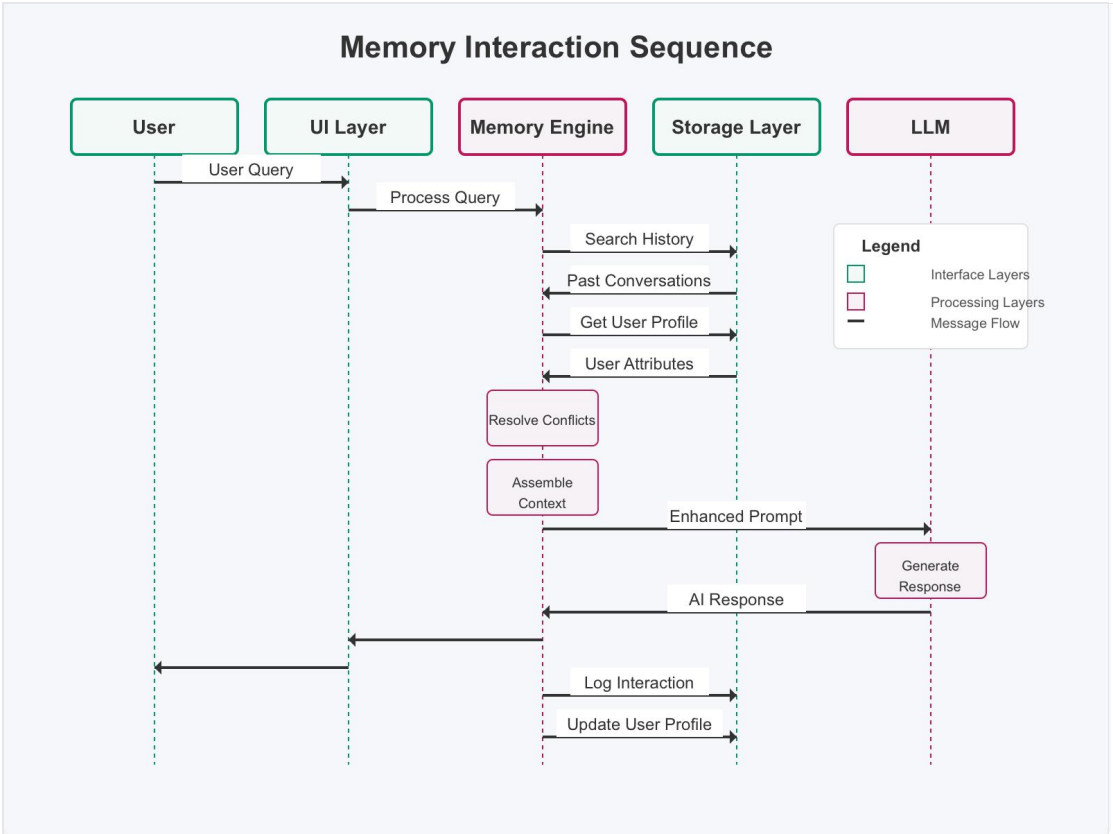


**Figure 2: Memory Types and Relationships**

User Profile Memory serves as the foundation, storing persistent facts about the user including preferences, demographic information, and important facts that remain relevant across all conversations. Complementing this, Conversation History maintains complete logs of past interactions, providing a rich source of context for future exchanges. The system also builds Extracted Knowledge by transforming unstructured conversation data into structured information. All of these feed into the Active Context—the currently relevant memories loaded specifically for each session, carefully selected to enhance the current exchange.

**The Memory Interaction Flow**

When a user interacts with a memory-enabled system like ChatGPT, memory flows through the system in a sophisticated sequence:



**Figure 3: Memory Sequence Diagram**

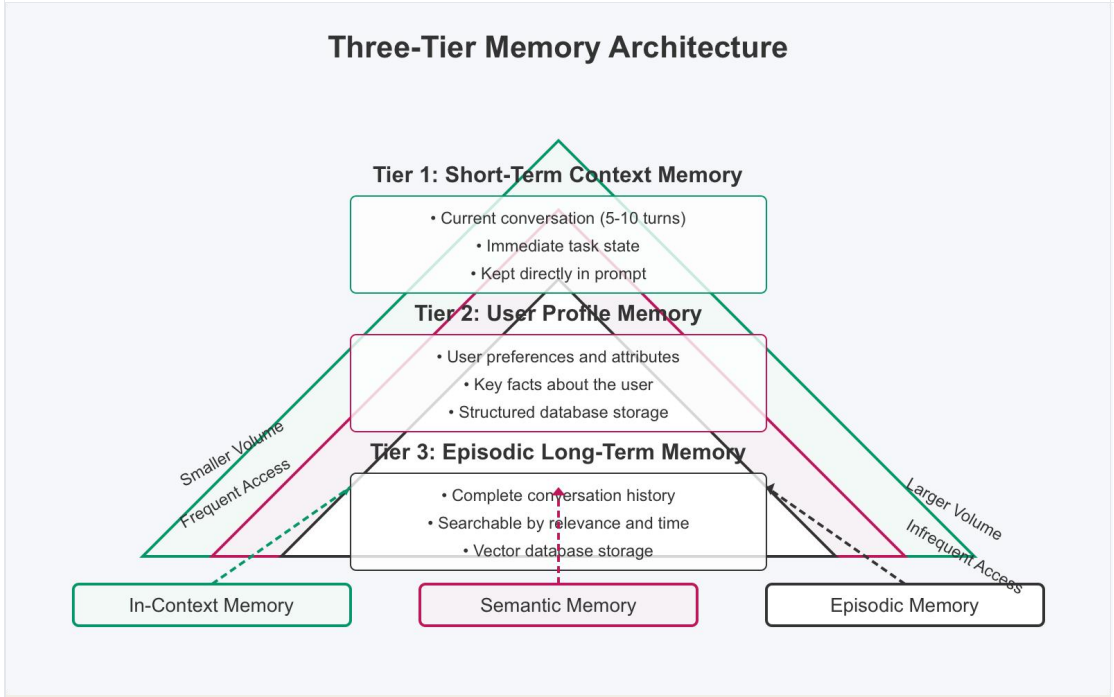
The process begins when a user sends a message, triggering the memory retrieval phase. The system immediately searches conversation history for relevant past interactions while simultaneously retrieving user profile information. It also works to identify any conflicting or outdated information that might need resolution. Next comes context assembly, where relevant memories are thoughtfully formatted and added to the prompt. The LLM then generates a response using this enhanced context, creating a more personalized interaction. After the exchange, the system updates its memory by logging the new conversation turn, updating the user profile with any new information, and resolving contradictions with existing memories.

### Building Your Own Memory System

Now, let's explore how you can implement similar memory capabilities in your own AI agent system, even without deep programming expertise.

### Designing Your Memory Architecture

The foundation of any effective memory system is a well-designed architecture. We recommend a three-tier approach that balances immediate context with long-term persistence:



**Figure 4: Three-Tier Memory Architecture**

The first tier, Short-Term Context Memory, maintains the immediate conversation flow. It typically contains the last 5-10 message turns kept directly in the prompt, lasting only for the current session. This gives your AI the immediate context needed for coherent exchanges.

The second tier, User Profile Memory, maintains persistent information about your users. This includes preferences, demographics, and important facts stored in a structured database that persists indefinitely until deletion. This profile grows richer with each interaction, allowing your AI to understand users better over time.

The third tier, Episodic Long-Term Memory, stores complete conversation history for retrieval. All past interactions are saved with timestamps and metadata in a vector database with search capabilities. These memories persist indefinitely, with optional archiving for older content, creating a rich history the AI can reference when needed.

Memory Management Workflows

The effectiveness of your memory system depends on well-designed workflows for storing and retrieving information:

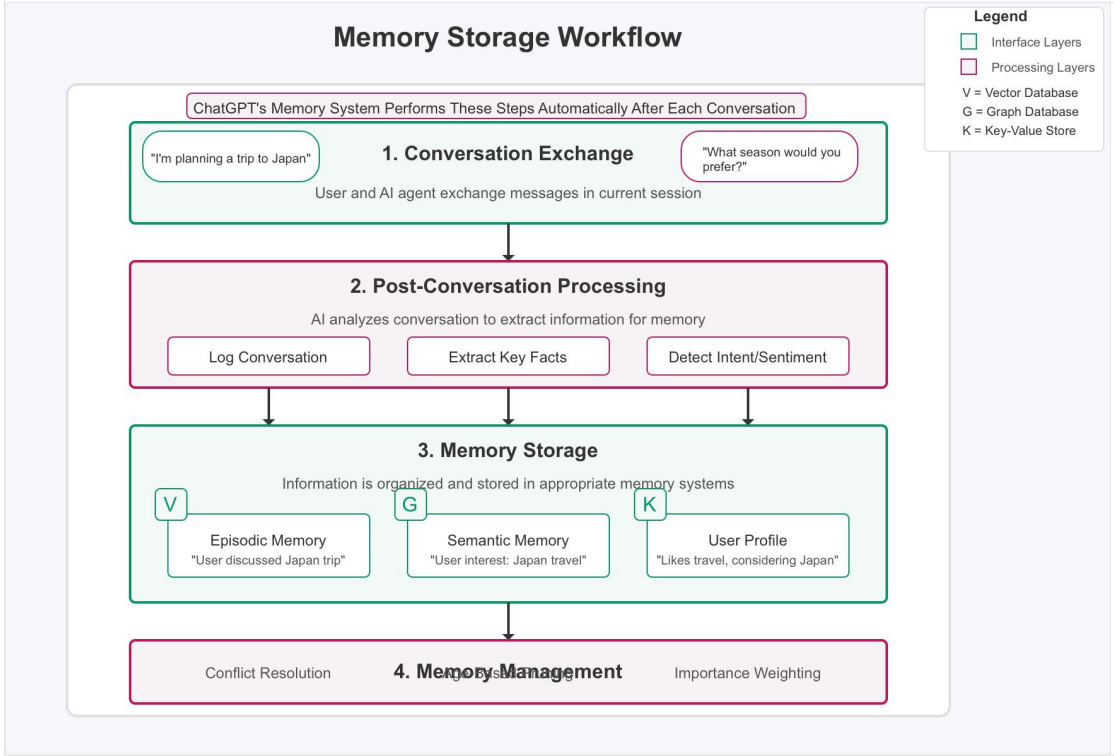
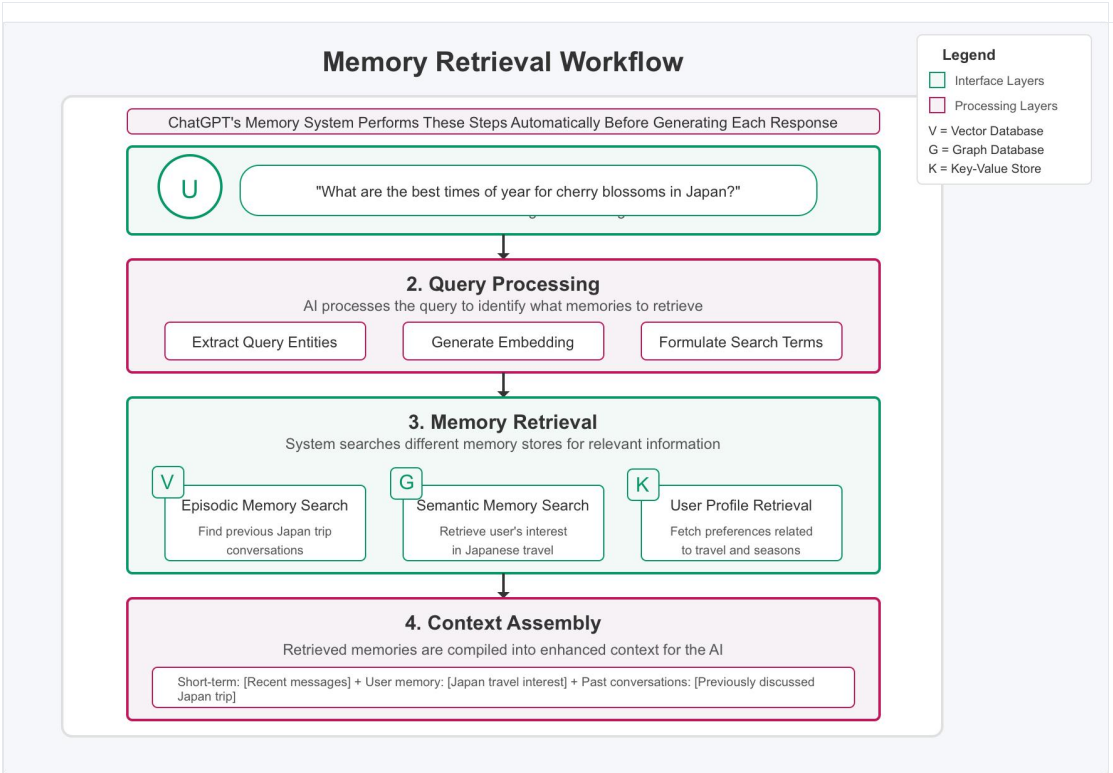


Figure 5: Memory Storage Process

The storage workflow begins when users and AI exchange messages. After each interaction, the system performs essential post-processing. It logs the full conversation to episodic memory while extracting key facts and preferences. These insights update the user profile with new information, with special attention to flagging contradictions with existing memories. This continuous refinement ensures the memory stays current and accurate over time.



**Figure 6: Memory Retrieval Process**

The retrieval workflow activates when a user sends a message. The system first pre-processes the query by searching episodic memory for relevant past conversations, retrieving current user profile information, and selecting the most relevant memories based on query context. It then assembles a comprehensive prompt combining short-term context from recent messages, relevant episodic memories, and user profile attributes. With this enriched context, the AI generates a response that feels informed by the complete relationship history.

## Technical Implementation Options

You don't need to build a memory system from scratch. Several existing technologies can serve as building blocks for your implementation.

**Implementation Tip**

Start with a simple memory architecture and gradually expand as your needs grow. Many projects can begin with just vector storage and basic retrieval before adding more complex features.

#### Vector Database Options

##### Pinecone

Cloud-based vector database with simple API that scales effortlessly. Ideal for production deployments with high reliability requirements.

##### Weaviate

Open-source vector search engine with rich semantic capabilities. Offers more flexibility for custom implementations.

##### Chroma

Lightweight embedding database designed specifically for retrieval-augmented generation. Perfect for smaller projects and quick prototyping.

#### Memory Management Frameworks

##### Mem0

Provides hybrid storage for user and agent memories, combining different storage types for optimal performance.



### **MemGPT/Letta**

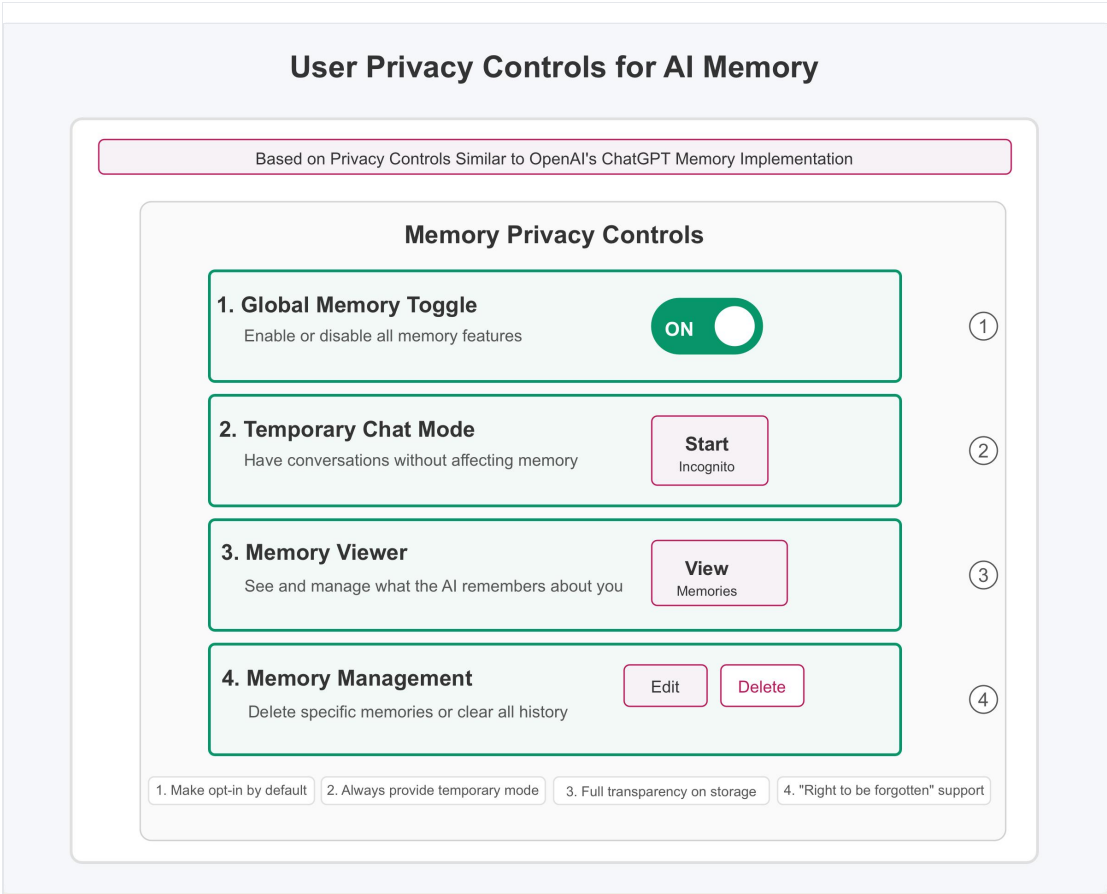
Offers hierarchical memory with tool-based access, giving AI agents more control over their memory systems.

### **LangChain Memory**

Includes conversation buffers and summary memory components that integrate smoothly with existing LangChain applications.

### **Privacy and Control Considerations**

Based on OpenAI's implementation, your memory system should incorporate robust privacy controls:



As your memory system grows, performance can degrade without proper optimization. A system that works perfectly with 100 memories might struggle with 10,000 or more.

As your system accumulates memories over time, performance optimization becomes increasingly important:

## Tiered Storage Strategy

Balance performance and cost with a multi-tiered approach:

- **Hot tier:** Recent and frequently accessed memories in fast storage
- **Warm tier:** Older but potentially relevant memories
- **Cold tier:** Archived memories rarely accessed

This approach ensures critical memories are always quickly accessible while optimizing resource allocation for less frequently used data.

## Memory Consolidation Strategy

Manage growing data volumes with intelligent consolidation:

- **Periodic summarization** of older conversations to distill their essence
- **Extraction of enduring facts** from ephemeral conversations
- **Resolution of conflicting information** across memory stores

These techniques maintain the most valuable information while reducing overall storage requirements.

## Retrieval Optimization Strategy

Ensure responsive performance as your memory grows:

- **Pre-computation of embedding vectors** during storage
- **Metadata filtering** before running expensive vector searches
- **Caching of frequently accessed memories** to reduce database load

These optimizations dramatically improve response times, especially in systems with large memory stores.

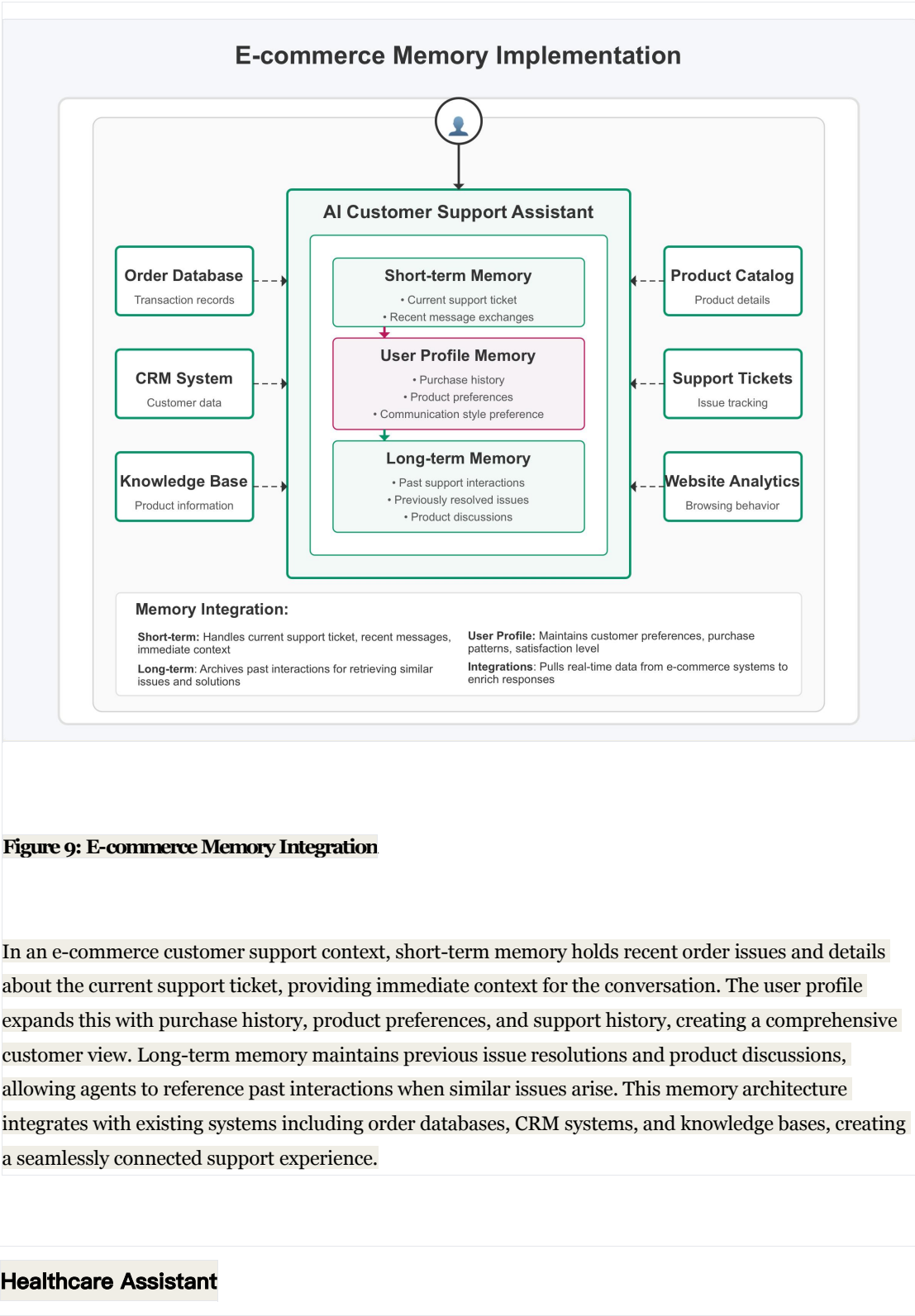
## Real-World Implementation Scenarios

### Industry Applications

Memory systems can be customized for specific industries, with each implementation prioritizing different types of information based on domain needs.

Let's examine how different organizations might implement memory in practical applications:

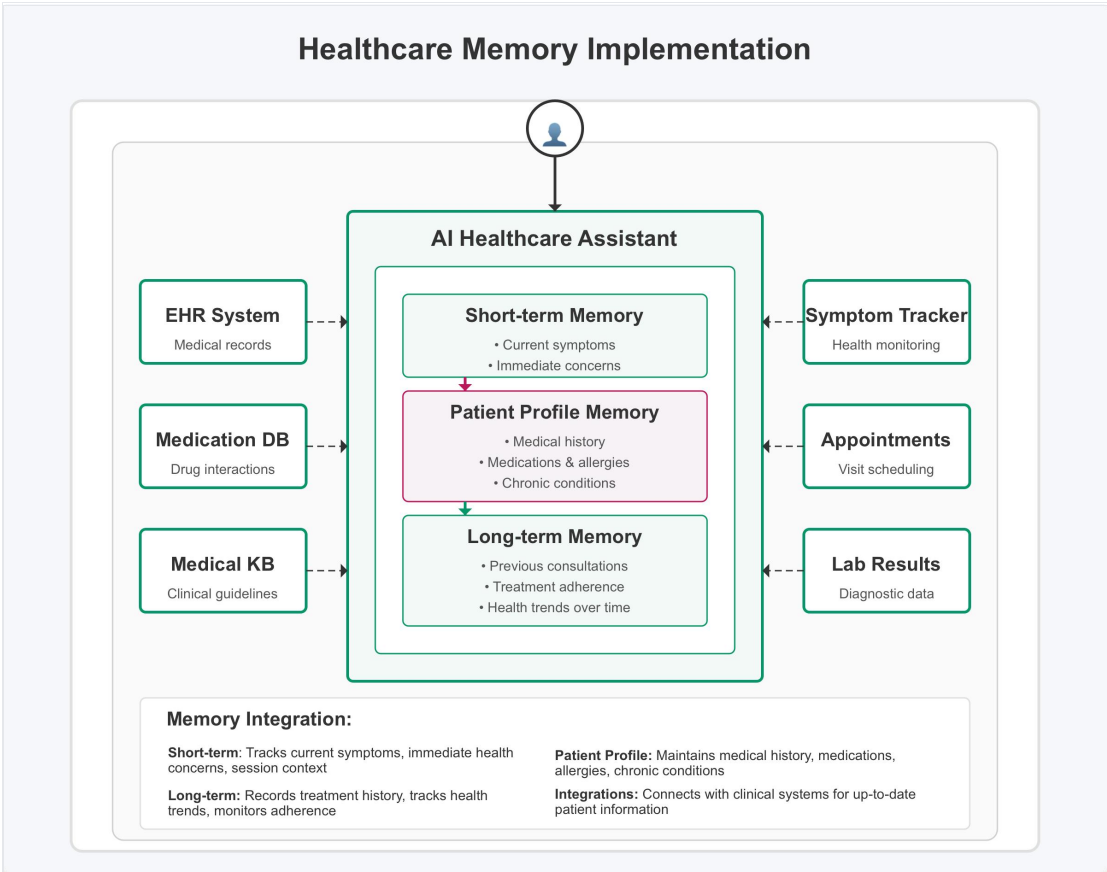
## E-commerce Customer Support



**Figure 9: E-commerce Memory Integration**

In an e-commerce customer support context, short-term memory holds recent order issues and details about the current support ticket, providing immediate context for the conversation. The user profile expands this with purchase history, product preferences, and support history, creating a comprehensive customer view. Long-term memory maintains previous issue resolutions and product discussions, allowing agents to reference past interactions when similar issues arise. This memory architecture integrates with existing systems including order databases, CRM systems, and knowledge bases, creating a seamlessly connected support experience.

**Healthcare Assistant**



**Figure 10: Healthcare Memory Integration**

For healthcare applications, short-term memory focuses on current symptoms and immediate health concerns, helping assistants address pressing patient needs. The patient profile contains critical information including medical history, medications, allergies, and chronic conditions. Long-term memory maintains records of previous consultations, tracks treatment adherence, and monitors health trends over time. These memory components integrate with healthcare systems including EHR platforms, medication databases, and symptom trackers, ensuring the assistant has a complete picture of patient health while maintaining compliance with healthcare regulations.

**Conclusion: The Future of AI Memory Systems**

**Key Takeaway**

OpenAI's memory upgrade validates what we've discussed throughout our memory series: truly intelligent AI agents require persistent, evolving memory to deliver personalized experiences.

The technical architecture we've outlined provides a blueprint for building your own memory-enabled system. You don't need to be a programmer to implement these concepts—many tools and frameworks now offer pre-built components you can integrate using configuration rather than coding.

## Getting Started with Memory Systems

### Define Your Memory Architecture

Start by mapping out the types of memory your system needs based on your specific use case requirements.

### Choose Your Storage Solutions

Select appropriate vector databases and other storage technologies that match your scale and performance needs.

### Implement Retrieval Mechanisms

Build efficient retrieval systems that can find the most relevant memories for each user interaction.

### Add Privacy Controls

Ensure your system includes robust privacy features that give users visibility and control over their data.

## Test and Optimize

Continuously test your memory system with real-world scenarios and optimize based on performance metrics.