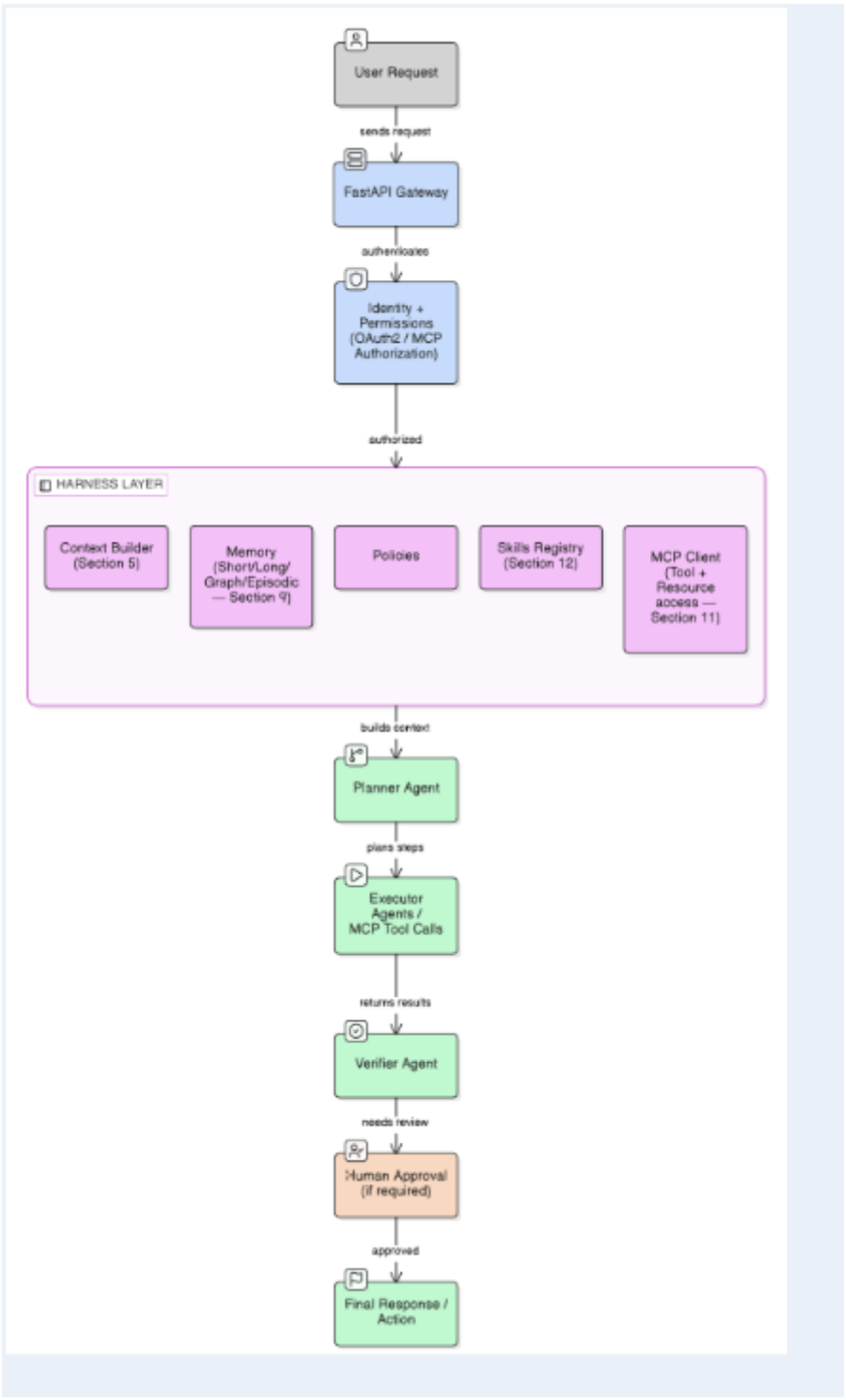
System Architecture

Architecture



1. User Request

The process begins when a user submits a request to the AI system. This request could be a question, instruction, or task that the user wants the system to perform. At this stage, the request

represents the user's intent, but it has not yet been verified or processed. Before the AI system can act on it, the request must pass through security and access control mechanisms.

2. API Gateway

The API Gateway acts as the main entry point between the user and the AI system. It receives incoming requests and manages how those requests are routed to the appropriate services. The gateway can also support functions such as request validation, rate limiting, logging, and traffic management. By providing a controlled entry point, it helps protect the underlying AI services from unauthorized or uncontrolled access.

3. Identity and Permissions

The Identity and Permissions layer is responsible for authenticating the user and determining what they are allowed to access or do. Authentication confirms **who the user is**, while authorization determines **what the user is permitted to do**. This layer ensures that the AI agent only accesses data, tools, and services that the requesting user has permission to use. It is a critical component for maintaining security, privacy, and organizational governance.

4. Harness Layer

The Harness Layer is the central orchestration environment in which the AI agent operates. Rather than allowing the model to directly access every available capability, the harness provides a structured environment containing the resources and controls required for the agent to perform its task. It brings together context, memory, policies, skills, and external tools, ensuring that the agent operates within a defined and governed framework.

5. Context Builder

The Context Builder prepares the information that the AI agent needs in order to understand and respond to the current request. It may combine the user's request with relevant documents, system instructions, retrieved knowledge, user information, and task-specific data. By constructing the right context, this component helps ensure that the agent has the information necessary to make informed decisions and generate relevant responses.

6. Memory

The Memory component allows the agent to maintain information beyond the immediate prompt. It may store details from the current interaction, previous conversations, completed tasks, or other relevant experiences. Memory helps the agent maintain continuity, avoid repeating work, and make decisions based on previous interactions. Depending on the architecture, memory may include short-term conversational memory, long-term memory, or task-specific memory.

7. Policies

Policies define the rules and boundaries that govern how the AI agent operates. These may include security requirements, organizational rules, privacy restrictions, compliance obligations, and limitations on what actions the agent can perform. The policy layer ensures that the agent's decisions and actions remain aligned with human-defined requirements rather than relying solely on the model's judgment.

8. Skills Registry

The Skills Registry contains the capabilities available to the AI agent. A skill may represent a specific function, workflow, tool, or specialized capability that the agent can use to complete a task. The registry allows the system to organize and manage these capabilities, making it easier for the agent to identify the appropriate skill when solving a problem. This approach also supports modularity, since new skills can be added without redesigning the entire agent.

9. MCP Client

The MCP Client connects the agent to external tools and resources using the Model Context Protocol (MCP). Through this component, the agent can access approved external services, APIs, databases, or applications. Rather than creating separate integrations for every tool, MCP provides a more standardized approach to connectivity. Importantly, access to these resources can still be controlled according to the user's permissions and organizational policies.

10. Planner Agent

Once the necessary context and resources have been assembled, the Planner Agent determines how the task should be completed. It analyzes the request, identifies the required steps, and creates a plan for execution. For simple requests, this plan may involve a single action. For more complex tasks, the planner may break the request into multiple steps and determine which tools, skills, or agents are required at each stage.

11. Agent and MCP Tool Calls

After creating a plan, the system executes the required steps through AI agents and MCP-enabled tools. The agent may call APIs, retrieve information, query databases, trigger workflows, or interact with external applications. Each action contributes toward completing the overall task. This stage is where the AI system moves beyond simply generating text and begins taking structured actions within the available environment.

12. Verifier Agent

The Verifier Agent checks the results produced during execution before they are returned to the user. It can assess whether the task was completed correctly, whether the response is consistent with the original request, and whether the output meets defined quality or policy requirements. This verification step adds an additional layer of reliability and can help detect errors before they reach the user.

13. Human Approval, If Required

For sensitive, high-impact, or irreversible actions, the workflow can include a human approval step. Instead of allowing the AI agent to automatically execute every action, the system pauses and requests approval from an authorized person. This human-in-the-loop approach is particularly important for decisions involving financial transactions, sensitive information, external communications, or other actions where human oversight is necessary.

14. Final Response or Action

Once the task has been successfully completed and, where necessary, approved by a human, the system produces the final response or executes the requested action. This is the outcome delivered back to the user. By the time the process reaches this stage, the request has passed through authentication, authorization, contextualization, planning, execution, verification, and potentially human oversight.

Overall Perspective

The diagram represents an evolution from a simple **“user prompt → LLM response”** model to a more mature **agentic AI architecture**. The AI agent sits within a controlled harness that provides context, memory, policies, skills, and access to external tools. A planning mechanism determines how tasks should be executed, verification checks the results, and human approval can be introduced when required.

Overall, this architecture is designed to create AI agents that are not only **capable and autonomous**, but also **secure, governed, modular, and accountable**.

Context Engineering Layer

Many failures are context failures, not model failures. This remains true, and is now the organizing principle of the discipline described in Section 5.

Context Builder Responsibilities

Assemble:

- Relevant memory
- Retrieved evidence
- Tool and skill availability
- User permissions
- Prior decisions
- Current workflow state
- Applicable policies

Rule: The best model with bad context still fails.

Memory Architecture

NEW IN 2026

Agent memory matured from "pick a vector database" into a benchmarked production discipline in 2026, with dedicated evaluation suites (LoCoMo (Long Conversation Memory)), MemBench, MemoryAgentBench, MemoryArena) and an ecosystem spanning roughly 20+ frameworks and vector stores across managed-cloud, self-hosted, and local-MCP hosting models.

Type	Storage	Layer	Purpose
Short-Term	Session	In-memory	Conversation state
Long-Term	Cross-session	Vector DB	Semantic recall
Graph Memory	Persistent	Graph DB	Relationships
Episodic	Persistent	DB	Prior tasks / outcomes
Procedural	Persistent	DB	Learned workflows
Policy Memory	Persistent	DB	Rules & restrictions
Audit Memory	Permanent	SQL	Logs & traceability

Three Patterns for Memory Control

- Pattern A — Context-resident: everything lives in the context window with compression. Simple, but caps out fast on long-running agents.

- Pattern B — Retrieval-augmented (workhorse pattern): working memory in-context, long-term records in a vector or structured store, injected each step. Recommended default, the engineering burden is manageable and the main challenge is retrieval quality.

- Pattern C — Tiered memory with learned control: multiple tiers (context, structured DB, vector store, cold archive) managed by a learned or prompted controller. Highest headroom, highest engineering cost — graduate to this only when data shows Pattern B is the bottleneck.

Best Practice

Memory entries should include:

- source
- timestamp
- confidence
- owner
- retention policy
- trust score

Known open problem across 2026 memory systems: selective forgetting. Most benchmarked systems handle retrieval and test-time learning reasonably well but still fail conspicuously at deciding what to evict. Budget explicit engineering time for eviction policy, not just ingestion.