AI Agents System Design

How to build scalable agentic systems?

Follow
Bhavishya Pandit

Repost

Introduction

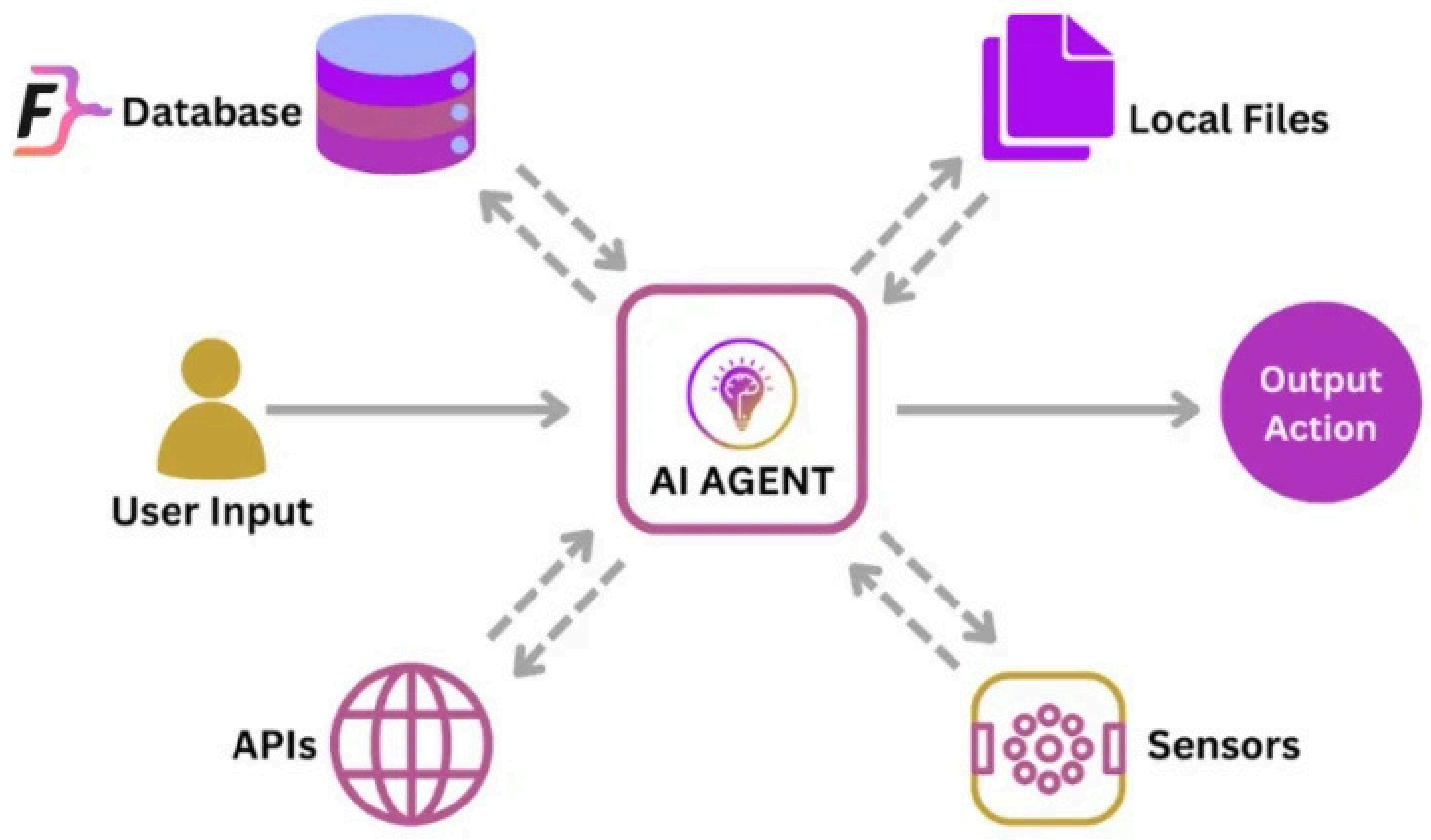
Modern AI apps are moving from single-turn Q&A models to systems that can reason, decide, and act.

All of this is possible due to **AI Agents**!

They enable AI to:

- Handle dynamic queries
- Fetch live data
- Take actions like updates or alerts

Source



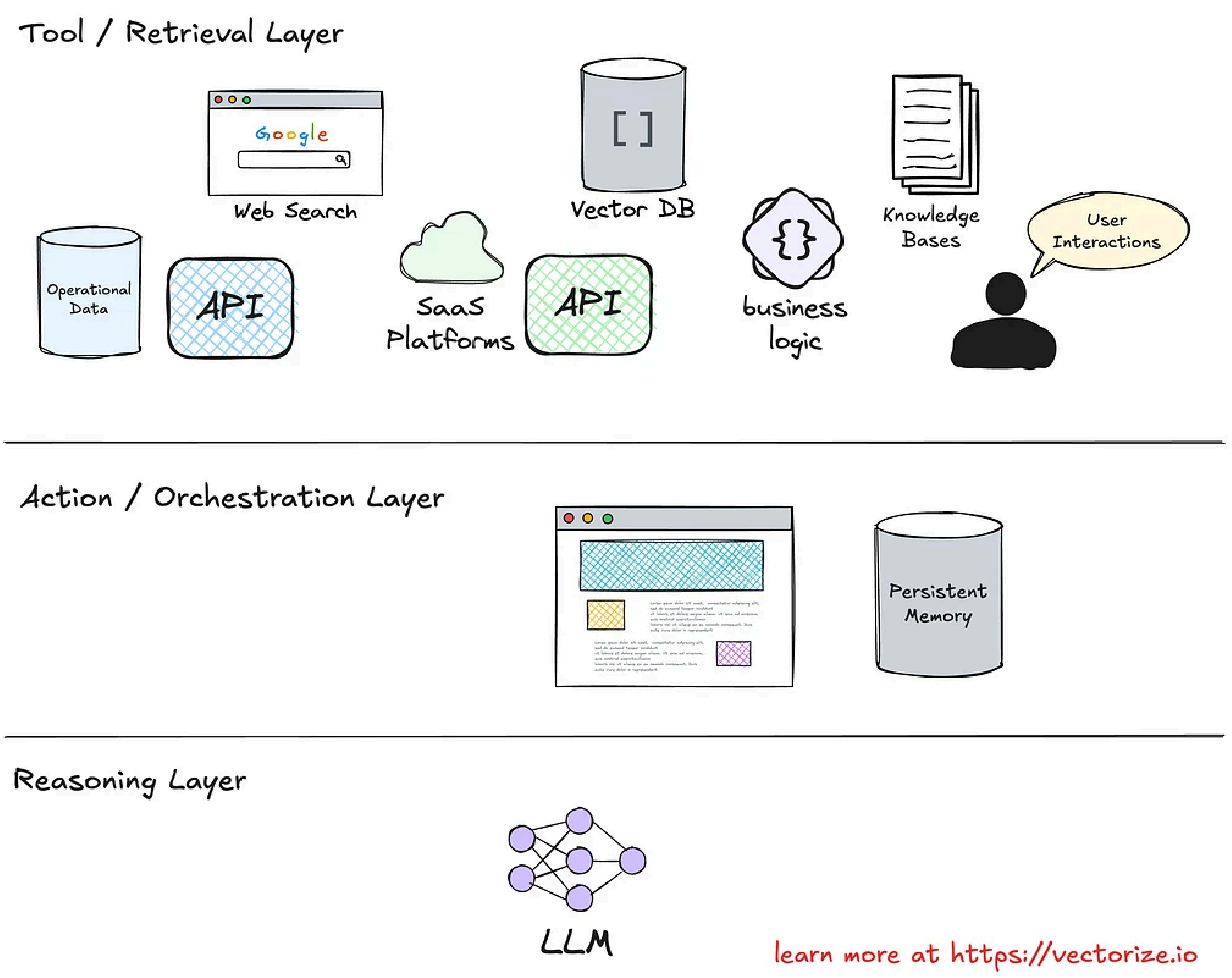
Agentic design is not one-size-fits-all. Let's explore how it works.

Follow
Bhavishya Pandit

Repost

Agentic System Architecture

A Agentic System is built with three layers like in this diagram:



Now lets learn the core components of agentic design -

Core Components of Agentic Design

Large Language Models (LLMs) are powerful, but by default, they're limited to what they were trained on.

Agentic systems extend this by adding:

1 Memory

Context awareness (e.g., conversation history)

2 Tool calling

Executing code, querying databases, etc.

3 Decision-making

Reasoning before acting

Now let's learn how each of these components can be built for scalability -

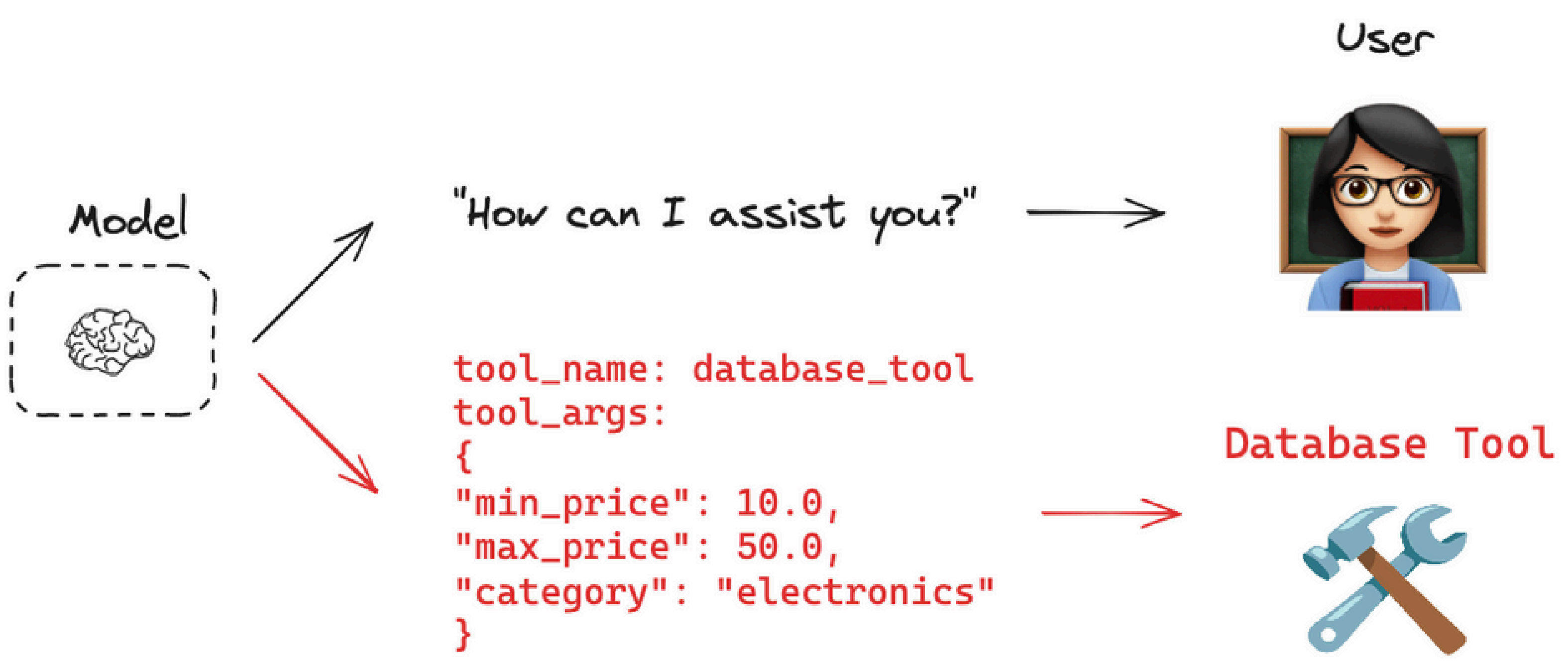
Follow
Bhavishya Pandit

Repost

Tool Calling

While basic tool calling works for prototypes, production systems require:

- 100-1000x more concurrent executions
- <500ms latency for real-time workflows
- Zero-downtime updates to tool inventory



```
tool_name: database_tool
tool_args:
{
"min_price": 10.0,
"max_price": 50.0,
"category": "electronics"
}
```

1. Async Task Queues

- Tools submit orders to Kafka/RabbitMQ
- Workers process first-come, first-served (or VIP priority)

2.Smart Throttling

- Reserve 20% capacity for critical tools (e.g., auth, payments)
- Batch non-urgent calls (e.g., nightly CRM updates)

3.Execution Pools

- Keep "hot" tools pre-warmed (user profile lookups)
- Isolate unstable tools to prevent cascading failures

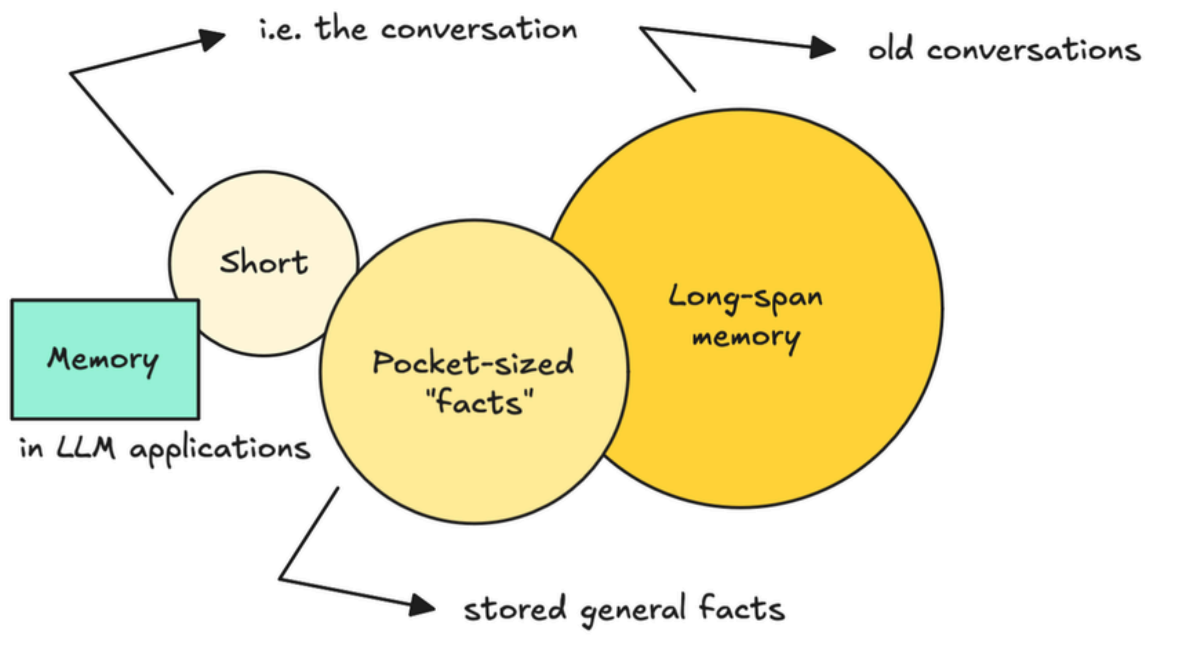
Follow
Bhavishya Pandit

Repost

Memory

Production AI agents require memory systems that balance speed, cost, and accuracy across:

- Short-term context (active conversation history)
- Long-term personalization (user preferences, historical interactions)
- Enterprise knowledge (product specs, compliance rules)



Real-Time Recall Layer - Use a fast in-memory store (like Redis) for real-time context—it keeps sessions snappy even during traffic spikes.

Persistent Memory Layer - A vector-capable database (like Pinecone) preserves nuanced user history without slowing down live interactions.

Governance Controls - Auto-expire old data and enforce access controls—this keeps systems lean and audit-ready.

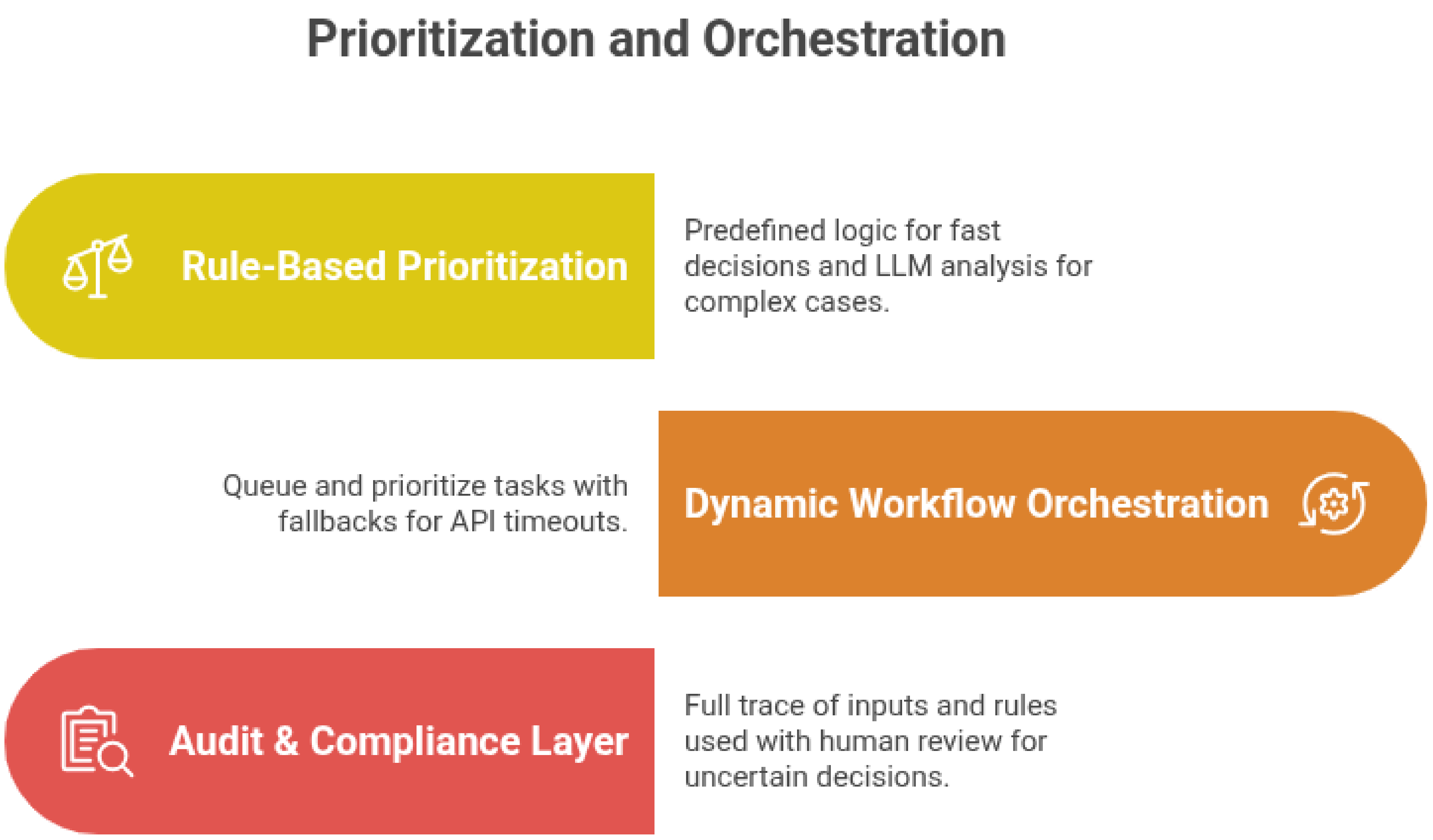
Follow
Bhavishya Pandit

Repost

Decision Making

AI agents must make fast, reliable decisions in production—balancing logic, business rules, and real-time data. Poorly designed systems lead to:

- Inconsistent outputs (e.g., approving invalid transactions)
- Bottlenecks (slow API calls blocking decisions)
- Compliance risks (violating regulations due to missing guardrails)



Prioritization and Orchestration

Rule-Based Prioritization

Predefined logic for fast decisions and LLM analysis for complex cases.

Dynamic Workflow Orchestration

Queue and prioritize tasks with fallbacks for API timeouts.

Audit & Compliance Layer

Full trace of inputs and rules used with human review for uncertain decisions.

Now let's explore some common agentic design patterns -

Follow
Bhavishya Pandit

Repost

1 Deterministic Chains

The most basic form of an agentic system is the deterministic chain.

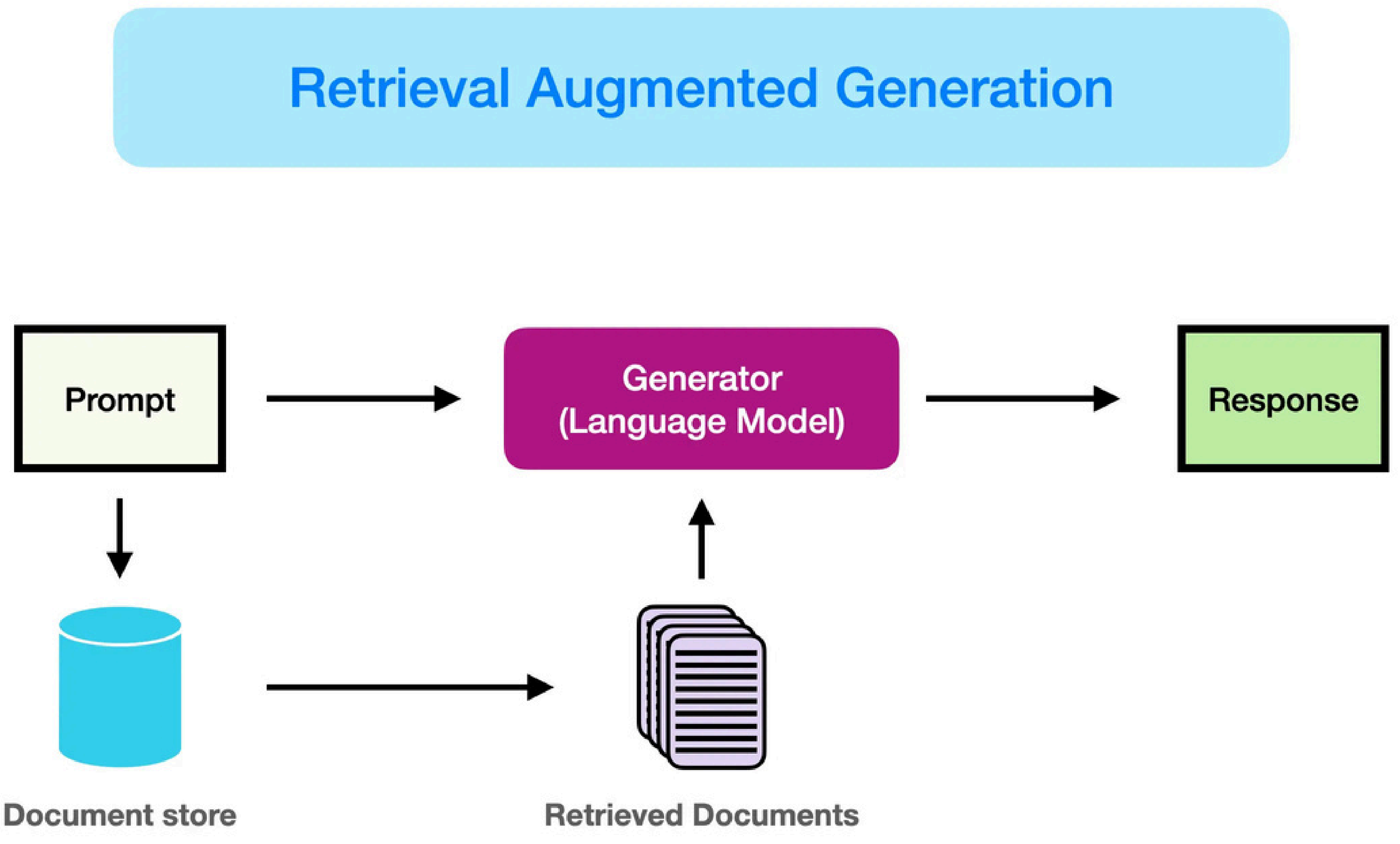
This design follows a fixed sequence of operations, where every step, tool call, and logic path is pre-defined by the developer.

✅ Every tool call happens in the same order

✅ The system behaves predictably for any given input

❌ No decisions, branching, or conditional logic

A common example is the Retrieval-Augmentation-Generation (RAG) workflow:



Follow
Bhavishya Pandit

Repost

2 Single-Agent Systems

In a single-agent system, one agent manages the entire flow — coordinating reasoning, tool usage, and responses in a dynamic and context-aware manner.

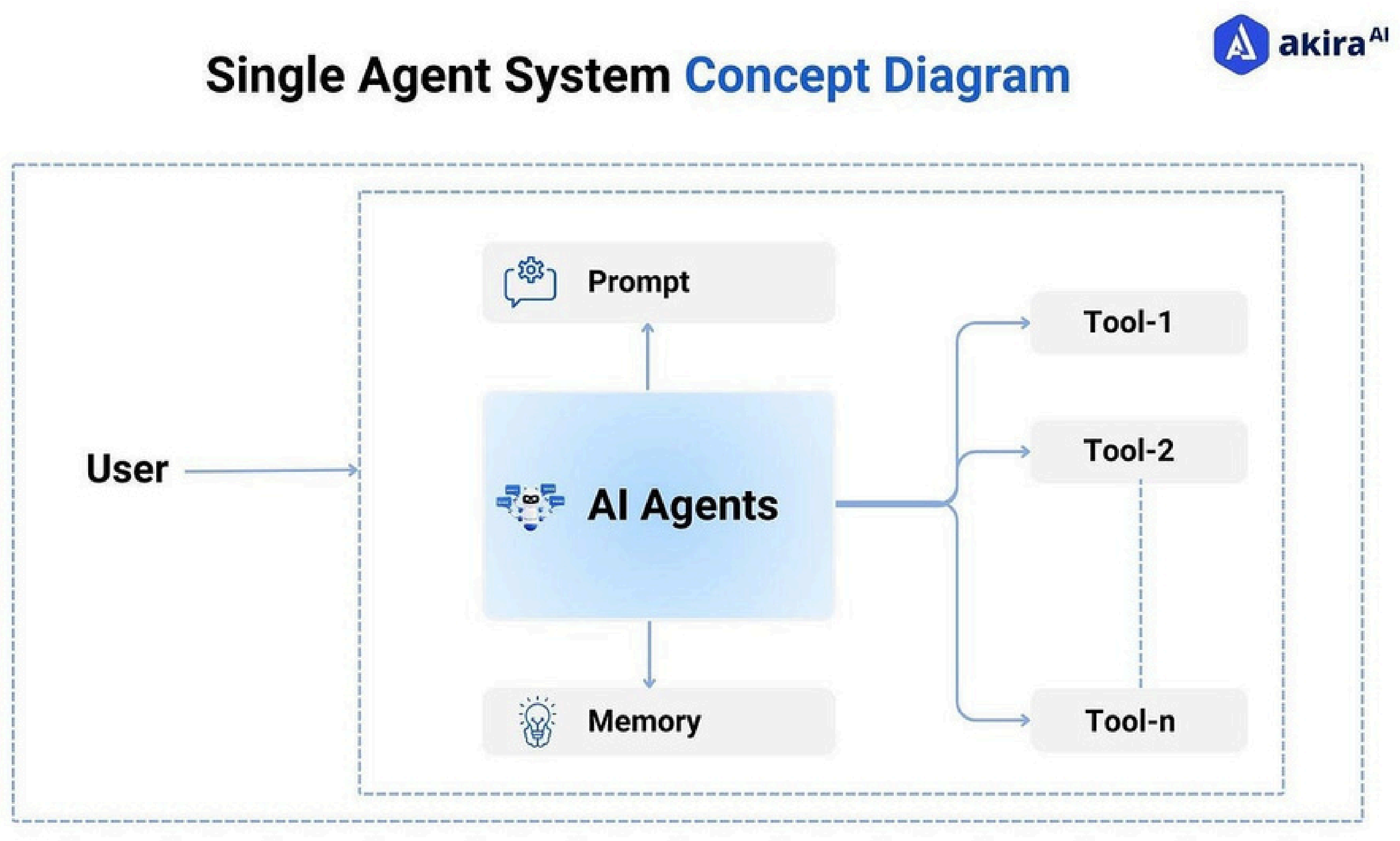
Unlike deterministic chains, the agent doesn't just follow a fixed script. It can:

✅ Accept inputs and relevant context (e.g., past conversations, user history)

✅ Decide when to call tools, and which tools to use

✅ Loop through reasoning steps — making multiple LLM/tool calls if needed to reach the desired outcome

Source



Follow
Bhavishya Pandit

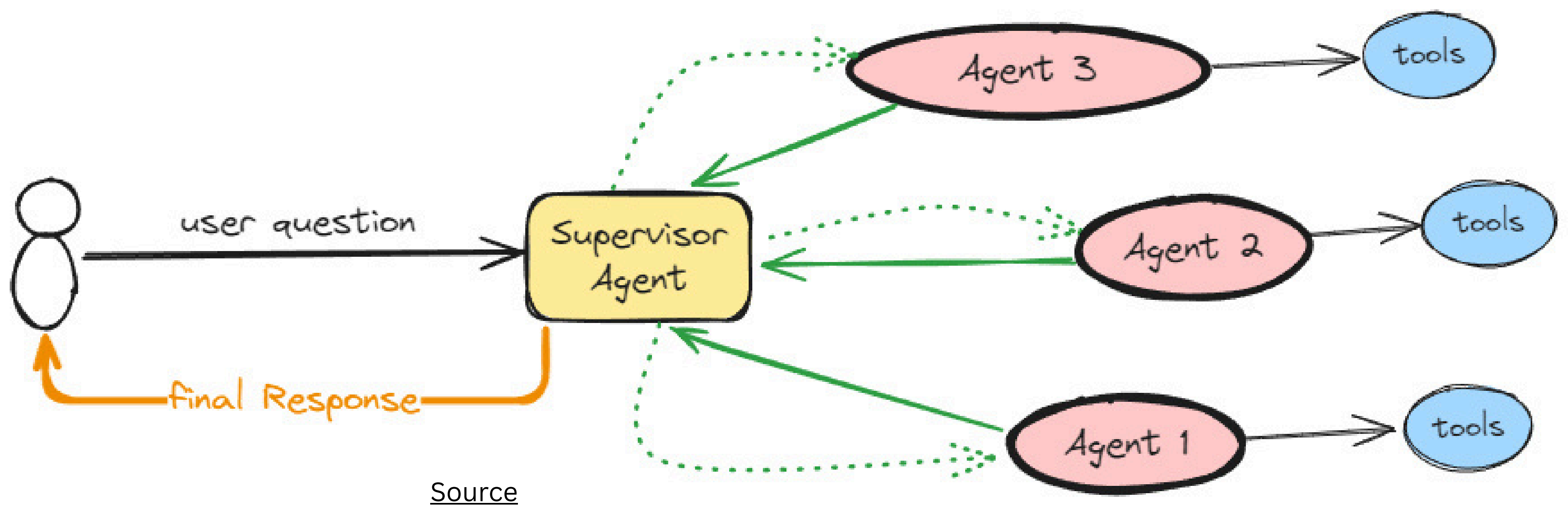
Repost

3 Multi-Agent Systems

In large-scale or cross-functional applications, a single agent may become overloaded or inefficient.

That's when a multi-agent system becomes essential.

Instead of one agent handling everything, the system is composed of multiple specialized agents, each responsible for a specific task or domain.



Source

Each agent:

✅ Focuses on a dedicated area (e.g., support, analytics, recommendations)

✅ Uses custom prompts, context, and tools relevant to its task

✅ Can collaborate or hand off to other agents when needed

Follow
Bhavishya Pandit

Repost

Choosing the Right Pattern

Not every AI application needs complex orchestration.

Choosing the right agent system design depends on your task complexity, domain diversity, and required flexibility.

Let's break it down:

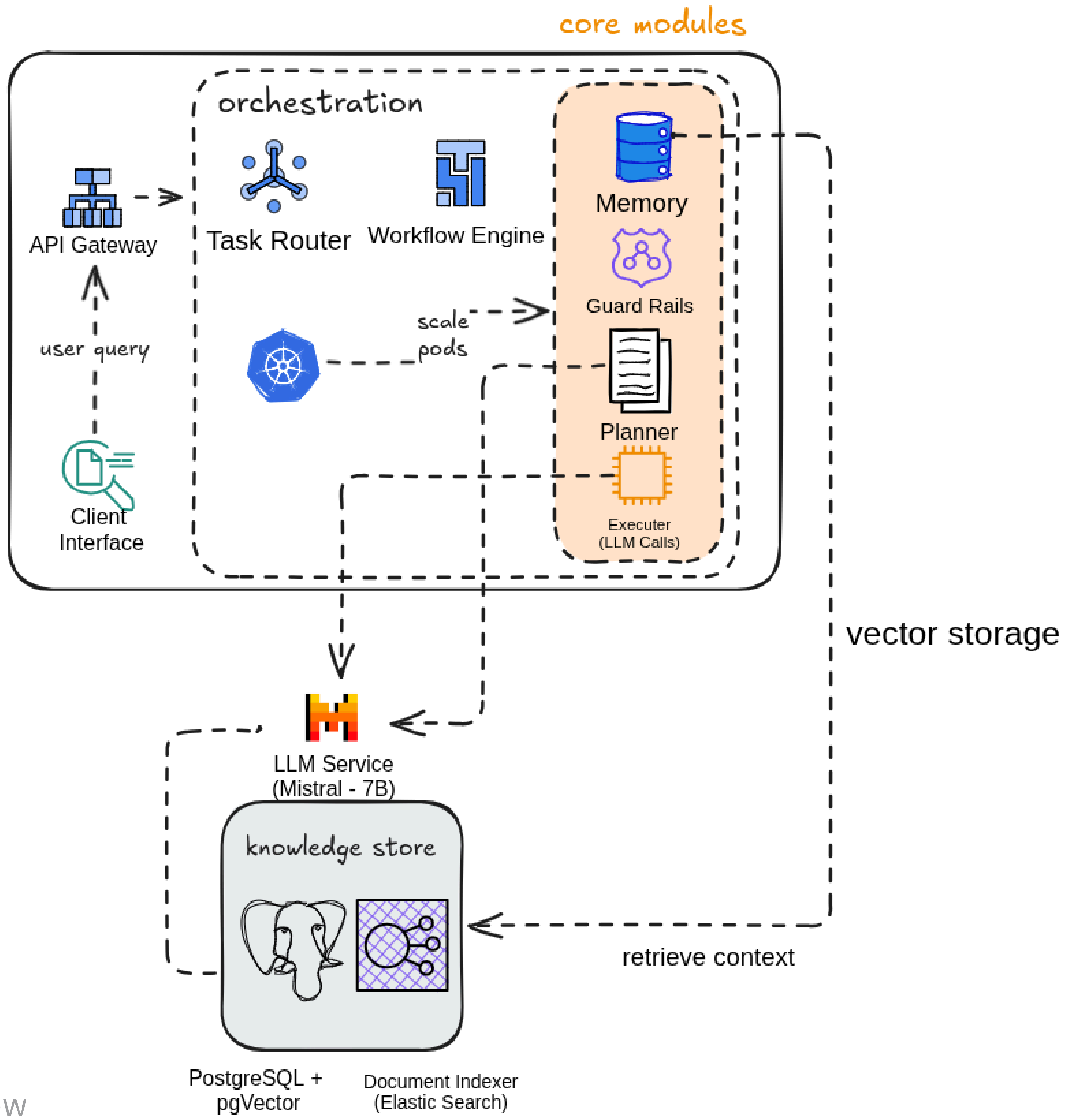
PATTERN	WHEN TO USE	EXAMPLE USECASES
Deterministic Chain	- Fixed, repeatable task flows - High auditability is required - Low latency and predictable output - Logic rarely changes	FAQ bots Compliance workflows Basic RAG pipelines
Single-Agent System	- Varied queries within a single domain - Needs reasoning or retries - Seeks flexibility without complexity - Moderate sophistication	Helpdesk assistants Internal AI tools Smart form-fillers
Multi-Agent System	- Covers diverse business domains - Specialized logic/context per task - Needs modular, scalable architecture - Clear role separation	Enterprise AI platforms AI copilots AI-powered CRMs

Follow
Bhavishya Pandit

Repost

Case Study: Bulding a Scalable Customer Support AI Agent

A leading e-commerce company sought to deploy an AI-powered chatbot to automate routine customer inquiries, reduce support costs, and deliver sub-second response times.



Follow
Bhavishya Pandit

Repost

Case Study: Bulding a Scalable Customer Support AI Agent

Key Points:

1.Model Selection:

Mistral 7B, fine-tuned on support transcripts, uses .

2. Retrieval-Augmented Generation (RAG):

Pinecone stores vectorized order/policy data, enabling similarity search for top-k context before LLM invocation.

3.Autoscaling & Orchestration:

Kubernetes HPA scales executors based on CPU/queue load, while Airflow schedules index rebuilds and Kafka decouples microservices.

4.Logging & Observability:

Elastic Stack logs structured JSON with correlation IDs, Prometheus/Grafana track metrics, and Jaeger visualizes OpenTelemetry traces.

Follow
Bhavishya Pandit

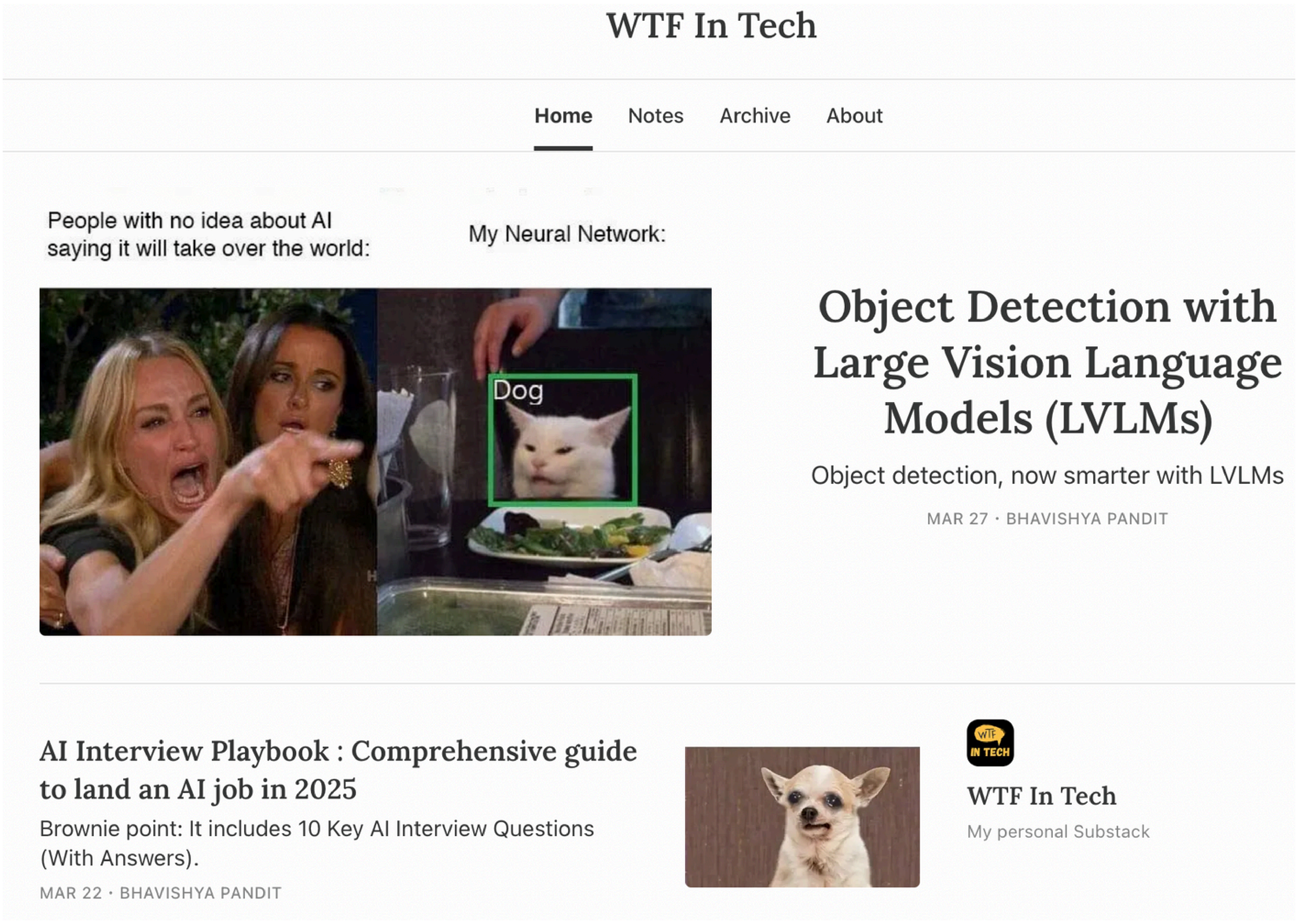
Repost

Stay Ahead with Our Tech Newsletter! 🚀

👉 **Join 1.1k+ leaders and professionals to stay ahead in GenAI!**
🔗 **https://bhavishyapandit9.substack.com/**

Join our newsletter for:

- Step-by-step guides to mastering complex topics
- Industry trends & innovations delivered straight to your inbox
- Actionable tips to enhance your skills and stay competitive
- Insights on cutting-edge AI & software development


WTF In Tech
Home
Notes
Archive
About
People with no idea about AI saying it will take over the world:
My Neural Network:
Dog

Object Detection with Large Vision Language Models (LVLMs)

Object detection, now smarter with LVLMs

MAR 27 · BHAVISHYA PANDIT

AI Interview Playbook : Comprehensive guide to land an AI job in 2025

Brownie point: It includes 10 Key AI Interview Questions (With Answers).

MAR 22 · BHAVISHYA PANDIT

WTF In Tech

My personal Substack

💡 Whether you're a developer, researcher, or tech enthusiast, this newsletter is your shortcut to staying informed and ahead of the curve.

Follow to stay updated on Generative AI

LIKE

COMMENT

REPOST