

ACCA7026

AI Agents and Agentic AI

Dr. Seena Joseph

Agentic AI

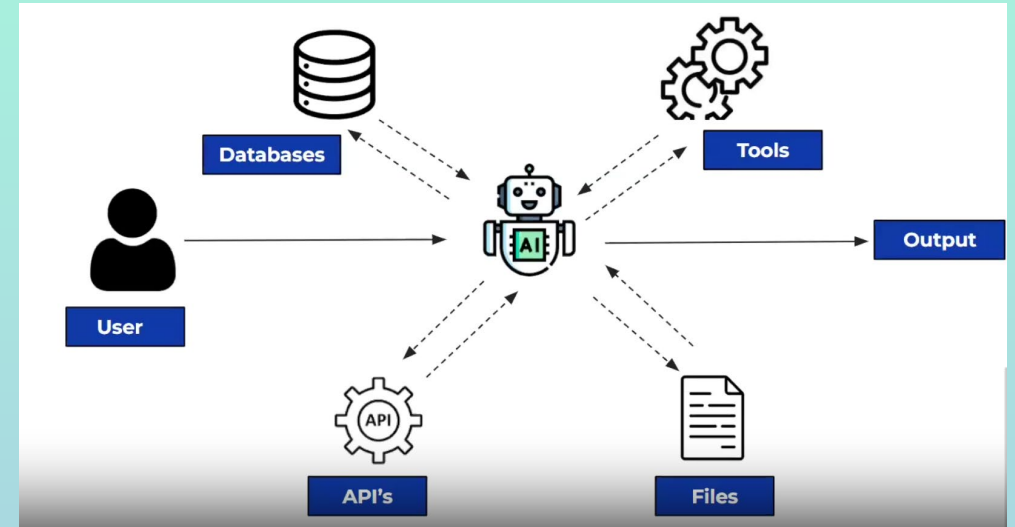
What Is an Agentic AI Framework?

AI agents are autonomous or semi-autonomous systems that use AI to perform tasks, make decisions, or solve problems, with little or no direct human intervention.

Reading the Diagram

The user sends a request to the agent at the centre. From there, the agent reaches out in whichever direction it needs to: a database, a tool, an API, or a file, before returning an output.

This same shape applies whether the agent is simple or advanced. At its core, it is just a language model wrapped in the ability to call things outside itself.



Agentic AI

LLMs Are Like a Brain

On their own, large language models are already very capable at language tasks: drafting an email, preparing a CV, planning an itinerary, or writing Python code.

The Gap

A plain LLM cannot act in the world. It cannot search for a job on your behalf, and it cannot submit an application for you; it can only produce text about doing so.

LLMs are like a brain without hands or legs: they can think, but on their own, they cannot act.

What Agentic AI Adds

Agentic AI gives that brain hands and legs: the ability to call tools, take actions, and carry out multi-step tasks in the real world, which is what the rest of this section covers.

Agentic AI

Tools: Giving Agents Hands

Agents act on the world through tools, for example: email servers (Gmail, Outlook), professional and booking platforms (LinkedIn, Booking.com), databases that store internal data such as user history or shopping patterns, APIs, and files.

Model Context Protocol (MCP)

As companies build more agents, each needing different tools, managing who can access what becomes unmanageable one integration at a time.

MCP solves this: every tool an agent might need sits behind an MCP server, which acts as a controller. An agent asks the server for access to a tool, and the server checks whether that agent is allowed to use it before any call is made.

The same way human developers rely on shared tools and standards, agents now need an equivalent, and MCP is the leading candidate.

Agentic AI

Agent-to-Agent Protocol (A2A)

An open, vendor-neutral communication standard, introduced by Google in 2025 and now governed by the Linux Foundation, that lets autonomous agents from different frameworks and developers discover, talk to, and delegate tasks to one another.

Key Concepts

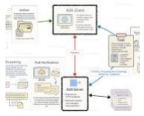
Agent Cards: standardised JSON documents, hosted at a known URL, that act like a digital business card advertising an agent's identity, functions and endpoints.

Client vs. server roles: a client agent requests work; a remote server agent listens for tasks, processes them, and returns results.

Opaque collaboration: agents work together without sharing internal memory, raw prompts, or direct access to each other's local tools.

 AI Overview

The **Agent2Agent (A2A)** protocol is an open, vendor-neutral communication standard that allows autonomous AI agents from different frameworks, platforms, and developers to discover, talk to, and delegate tasks to one another. Google originally introduced A2A in April 2025, and governance was later transferred to the [Linux Foundation Agent2Agent Project](#).



Key Concepts

- **Agent Cards:** Standardized JSON documents hosted at a known URL on an agent's server that advertise its identity, functions, communication endpoints, and authentication rules (acting like a digital business card).  YouTube · Google Cloud... +1
- **Client vs. Server Roles:** A *client agent* requests work to be performed, while a *remote server agent* listens for tasks, processes them, and returns results.  CodiLime +1
- **Opaque Collaboration:** Agents collaborate without sharing internal memory, raw prompts, or direct access to each other's local tools.  GitHub +1
- **Underlying Tech Stack:** Built on standard web technologies including HTTP(S), JSON-RPC 2.0, gRPC, and REST, with support for Server-Sent Events (SSE) for streaming updates.  Mastra +2
- **Security & Auth:** Supports enterprise security schemes such as Transport Layer

Agentic AI

Introduction to Agentic Frameworks

An agentic framework is a conceptual blueprint for designing and implementing AI agents: how these autonomous systems operate to achieve specific goals.

Agentic frameworks leverage LLMs to enhance an agent's understanding of natural language, letting it process complex queries, generate human-like responses, and better interact with users and context.



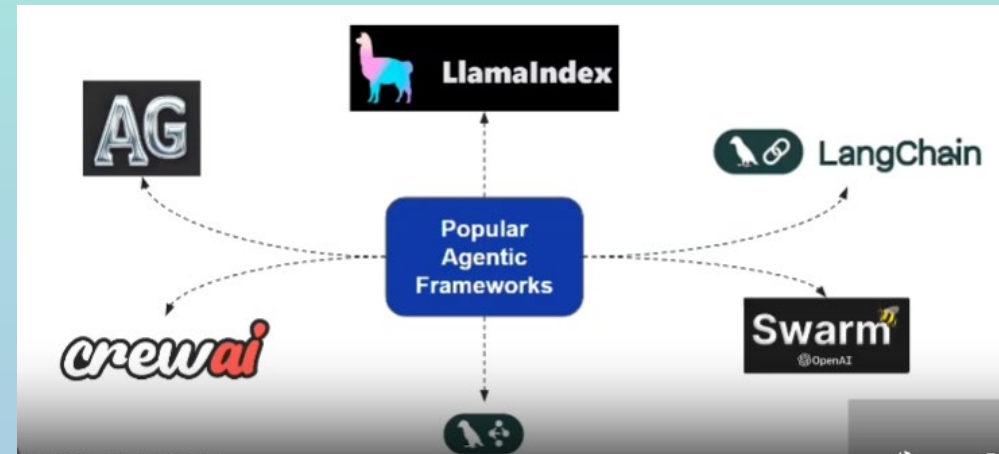
Agentic AI

[LangGraph: Agent Orchestration Framework for Reliable AI Agents](#)

The Framework Landscape

No single framework dominates. AutoGen, LlamaIndex, LangChain, CrewAI and Swarm each take a different approach to the same underlying problem: how do you reliably get an LLM to plan, call tools, and act over multiple steps?

The next two slides introduce the frameworks most relevant to this module.



Agentic AI

[LangGraph: Agent Orchestration Framework for Reliable AI Agents](#)

LangChain

Provides high-level building blocks for creating agents: chaining together LLM calls, tool use, and data sources into a working application.

LangGraph

The lower-level orchestration engine underneath: it represents an agent's logic as a graph of steps, with built-in state, memory and checkpointing, so multi-step agent behaviour is reliable and inspectable. (The name refers to this graph of steps, not to knowledge graphs.)

LlamaIndex

Combines LLMs with index-based search, giving an agent efficient access to large external datasets and contextually relevant answers grounded in that data.

Agentic AI

Swarm

Uses decentralised, collective-behaviour-style algorithms, letting multiple agents collaborate and optimise problem-solving and decision-making between themselves.

CrewAI

Emphasises collaboration among role-based AI agents, suited to applications needing teamwork and shared decision-making, and the framework behind this module's own hands-on case study.

AutoGen

Automates much of the plumbing of building AI applications: user-friendly tools that integrate LLMs and streamline development with minimal coding.

Agentic AI

No-Code Agentic AI: n8n

No code agentic AI framework – N8N

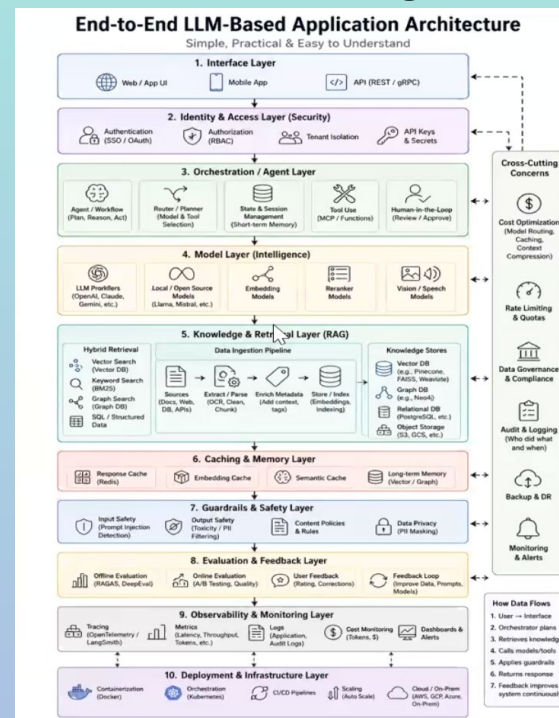
[AI Workflow Automation Platform - n8n](#)

n8n lets you build agentic workflows visually, by connecting nodes on a canvas, rather than writing code. It is a practical option for teams who want to wire up agents, tools and APIs quickly without a full development setup.

Agentic AI

End-to-End LLM-Based Application Architecture

Everything covered so far, the model, tools, MCP, A2A, and the frameworks that tie them together, has a place in a real production system. This diagram maps the whole stack end to end; the next slide walks through each of its ten layers.



Agentic AI

The Ten Layers, Top to Bottom

1. Interface layer, how the user reaches the system: web or app UI, mobile app, or a REST/gRPC API.
2. Identity & access layer (security): authentication, authorisation, tenant isolation, and API keys/secrets, who is allowed to do what.
3. Orchestration / agent layer, where LangChain or LangGraph run the show: planning, routing between models and tools, managing session state, and deciding when a human needs to review or approve a step.
4. Model layer (intelligence), the actual brain: LLM providers such as OpenAI, Claude or Gemini, local/open-source models, plus embedding, reranker and vision/speech models.
5. Knowledge & retrieval layer (RAG), how information is found and fed to the model: hybrid retrieval (vector, keyword, graph, SQL), a data ingestion pipeline, and knowledge stores.
6. Caching & memory layer: response, embedding and semantic caches, plus long-term memory, so the system is not repeating the same work.
7. Guardrails & safety layer: input safety (prompt injection detection), output safety (toxicity/PII filtering), content policies, and data privacy.
8. Evaluation & feedback layer: offline evaluation (RAGAS, DeepEval), online evaluation (A/B testing), user feedback, and the feedback loop that improves the system.
9. Observability & monitoring layer: tracing, metrics (latency, throughput, tokens), logs, cost monitoring, and dashboards & alerts.
10. Deployment & infrastructure layer: containerisation, orchestration, CI/CD, auto-scaling, and where it runs (cloud or on-prem).

Cross-Cutting Concerns

Running alongside every layer: cost optimisation, rate limiting & quotas, data governance & compliance, audit & logging, backup & disaster recovery, and monitoring & alerts.

How Data Flows

User reaches the interface, the orchestrator plans, retrieves knowledge, calls models/tools, applies guardrails, returns a response, and feedback improves the system continuously.

Agentic AI

Hands-On Case Study: Context

Most LLM assistants can chat, summarise and answer questions, but cannot act on your data.

This case study asks: can an LLM-powered agent take a plain-English question, write the pandas code needed to answer it, and run that code directly on a real dataset?

We build a Data Analysis Agent using the LangChain framework on top of an LLM, so natural-language requests are converted into working code without the user writing any code themselves.

Case Study Objective

Develop an agent that can load a dataset, answer descriptive, comparative and correlation questions about it, and generate a chart on request, entirely through conversation.

Agentic AI

Hands-On Case Study: What We Built

Environment: a dedicated conda environment, with LangChain, CrewAI and the Gemini API client installed.

Model: Google Gemini used as the LLM core, a free-tier alternative to a paid OpenAI key, wired in through LangChain's ChatGoogleGenerativeAI.

Agent: LangChain's pandas dataframe agent, given a real dataset and allowed to write and run its own pandas code to answer questions.

Demo questions: "What is the average age, and what percentage have heart disease?" and "Which variable correlates most strongly with heart disease?"

Connecting back to today's six components: Perception (reading the dataset), Planning & reasoning (deciding what code answers the question), Action & tool use (running that code), and Memory (the running conversation) all work together, live, in this one small agent.

Try it yourself: full step-by-step build instructions are in the companion tutorial, "Building a Free, Gemini-Powered Data Analysis Agent with LangChain."