Building AI Agents with LLMs:

A Practical Introduction

Agenda

1. Introduction to AI Agents
2. Architecture of AI Agents
    - Retrieval
    - Tools(with demo)
    - Memory
3. Agent Workflows(with demo)
    - Prompt Chaining
    - Routing
    - Parallelism(Concurrent Agent Execution)

1. Introduction to AI Agents

An **AI Agent** is an autonomous or semi-autonomous system that leverages a large language model (LLM) to understand user intent, reason through possible actions, and execute tasks by integrating external tools (e.g., APIs or custom functions), accessing stored knowledge (memory), and retrieving external information.

These agents go beyond basic question answering by chaining logic, invoking tools when needed, and maintaining context over time.

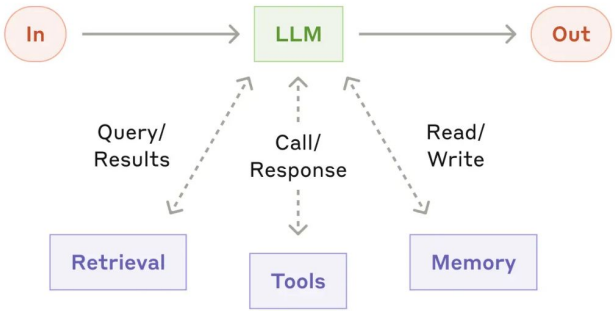
Cont..

Example:

1. **GitHub Copilot:** Acts as an agent by assisting developers. It understands code context and suggests code, queries documentation, and helps debug, behaving intelligently within an IDE.
2. **AI customer assistants:** These are agents deployed on websites or apps that not only chat but also perform actions like checking order status, booking appointments, or updating user info, often using APIs behind the scenes.

They all go beyond static Q&A—they reason, act, and remember, which is what defines an AI agent.

2. Architecture of AI Agents Built on LLMs



2. Architecture of AI Agents Built on LLMs

The **architecture of AI agents** refers to the structural design that defines how an AI agent senses, processes, decides, and acts within its environment.

An **AI agent built on an LLM** is typically modular, meaning it consists of interchangeable and cooperative components that extend the LLM's native capabilities.

Core Components of the LLM-Based AI Agent Architecture

1. **LLM Backbone (Central Reasoning Engine)**
2. **Retrieval Module**
3. **Tools / Plugins / Function Calling**
4. **Memory**

1. **LLM Backbone (Central Reasoning Engine)**

Acts as the **brain** of the agent.

Handles:

- Natural language understanding
- Reasoning
- Decision making
- Response generation

Example: OpenAI GPT-4, DeepSeek, Claude, Gemini…

Retrieval

Retrieval is the process of **finding relevant information from a large dataset or knowledge base** to provide the AI model with context before or during generating an answer.

Retrieval refers to the AI agent's ability to **look up information** from an **external knowledge base**, such as a database, search engine, or document collection, **on demand** rather than relying only on what it was trained on.

How it works:

- The AI receives a user question.
- It formulates a search query or fetches relevant documents from an external source (e.g., web, vector database).
- It uses the retrieved content to generate a more accurate and informed answer.

Tools

These are external functions/APIs the LLM can call to do things it can't do on its own — like fetch live weather data, do calculations, or interact with databases.

Tools are **external functions or APIs** that an AI can *call* to get specific information or perform actions beyond just generating text.

How they work:

The AI decides when it needs to call a tool based on the user's query. The tool is invoked with structured parameters, and the AI then uses the tool's output to craft a better answer.

User say: "Show me the current exchange rate of USD to ETB (Ethiopian Birr) from Commercial Bank of Ethiopia."

Why use a tool here?

- Because the AI itself **does not have live access** to current exchange rates.
- The tool (API) provides **real-time, accurate data**.
- This makes the AI's answer reliable and up-to-date.

Memory

This is where the agent stores past interactions or facts for continuity and personalization across sessions.

Memory in an AI agent refers to its ability to **remember past interactions** over **long periods of time**, beyond the current session.

There are two types:

- **Short-term memory** (within the current conversation)
- **Long-term memory** (Remembering that you like vegan recipes even weeks later)

Scenario

Let’s say we are building an AI **financial advisor assistant**.

User say: “I want to invest ETB50,000. What are my options?”

Memory: Recalls you are risk-averse(don't like **risky investments)** and interested in tech stocks.
Retrieval: Pulls up the latest stock market data and trends.
Tool: Uses a calculator tool to model expected returns over 5 years.
Response: Gives a personalized investment plan, with updated data and projections.

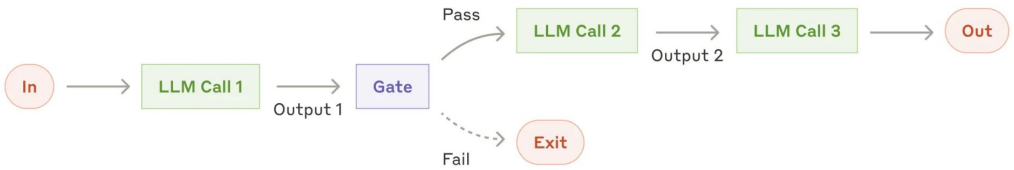
Agent Workflows

1. Prompt Chaining
2. Routing
3. Parallelism(Concurrent Agent Execution)

Prompt Chaining

Prompt chaining is the technique of passing the output of one prompt as the input to another prompt to build a multi-step reasoning or task execution pipeline.

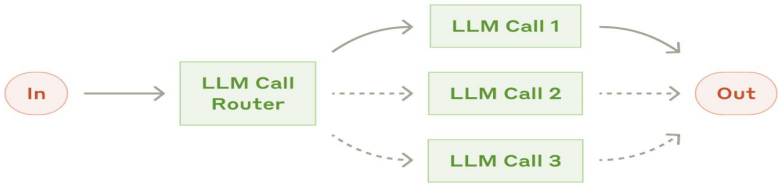
Example: First, get the destination from the user → then search for hotels → then get flight options → then summarize the best package.



Routing (Dynamic Prompt Routing)

Routing is the process of **dynamically deciding which prompt, model, or agent to use next** based on the input, context, or intermediate results.

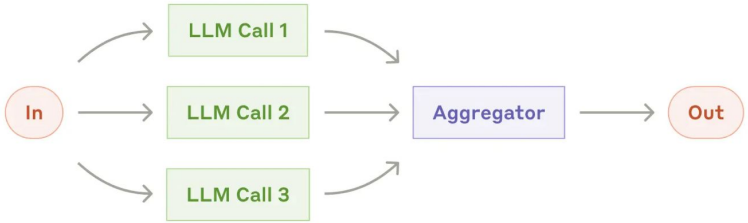
Example: If the user asks about flights → route to flight API agent. If the user asks about visa requirements → route to documentation agent.



Parallelism (Concurrent Agent Execution

Parallelism in agents means **executing multiple prompts, tools, or tasks at the same time** instead of sequentially, to save time.

Example: Search for hotels, flights, and local attractions **simultaneously** to save time.



Flow

1. **Input**: User provides a question or command.
2. **LLM**: Processes input and decides if it needs to retrieve info, call a tool, or use memory.
3. **Decision**: Internally, the LLM determines how to handle the request.
4. **Use Module**: It interacts with the right module (retrieval, tool, or memory).
5. **Response**: Final output is generated and returned to the user.