

---

# Agentic AI & LangGraph Essentials

Dipanjan (DJ) Sarkar

*How do we perform tasks?*

*We think, observe and we act.*

# Reasoning + Actions

# Example of Reasoning + Action

Decision-making process of choosing to buy or sell a specific stock in the market



This would depend on obtaining additional information about such stocks (**performance, news**) before taking action (**BUY / SELL**).

# Example of Reasoning + Action - Buy or Sell a Stock

Task: I want to see if I should purchase Nvidia (NVDA) or Intel (INTC) stock



# Example of Reasoning + Action - Buy or Sell a Stock

**Task:** I want to see if I should purchase Nvidia (NVDA) or Intel (INTC) stock

## Response

NVDA appears to be experiencing rapid growth, making it a potentially attractive BUY for growth investors. Meanwhile, INTC has seen a decline in its stock price, which may present a viable buying opportunity for value investors looking for long-term holdings; however, it might be wise to avoid it for the time being.

# What is Reasoning?

Reasoning = Thinking + Planning + Observing

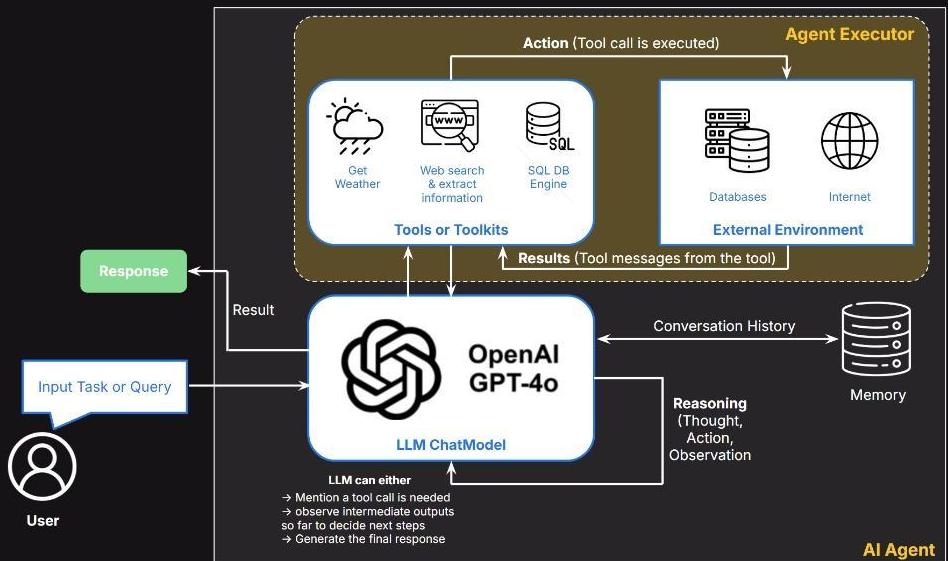
# What is Action?

Interact with external environment:

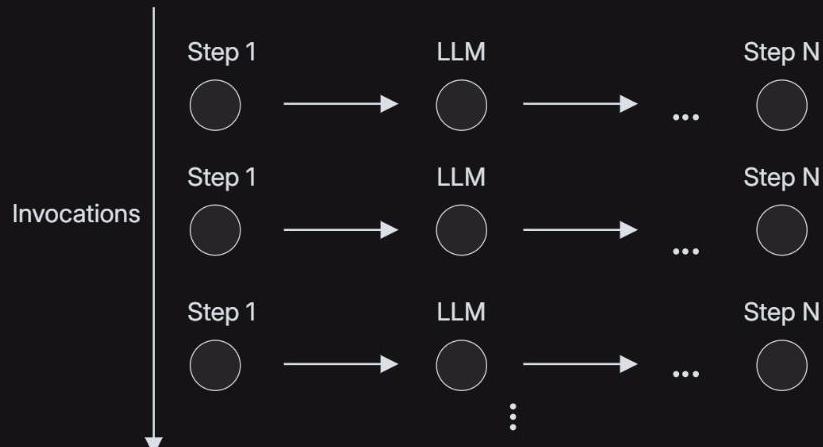
Use tools like search, stock APIs, check news, query databases etc.

# Key Components of an AI Agent

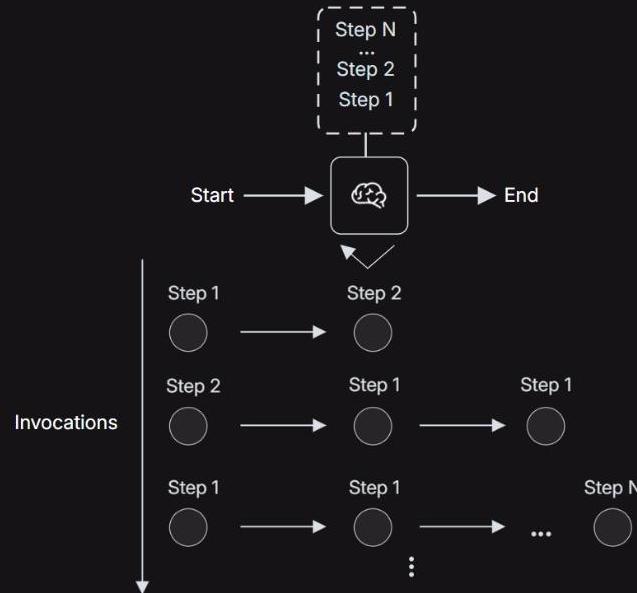
- LLM (Reasoning Engine)
- Planning Module (ReAct or Custom)
- Tools (Actions)
- External Knowledge Bases
- Memory



# AI Workflows vs. AI Agents



AI workflows always execute the same flow



Agentic AI systems rely on the LLM to control the flow

# Why LangGraph?

LangGraph is designed for building agentic applications with some core principles:

---

- **Controllability:**

Offers low-level control which increases reliability in agentic systems.

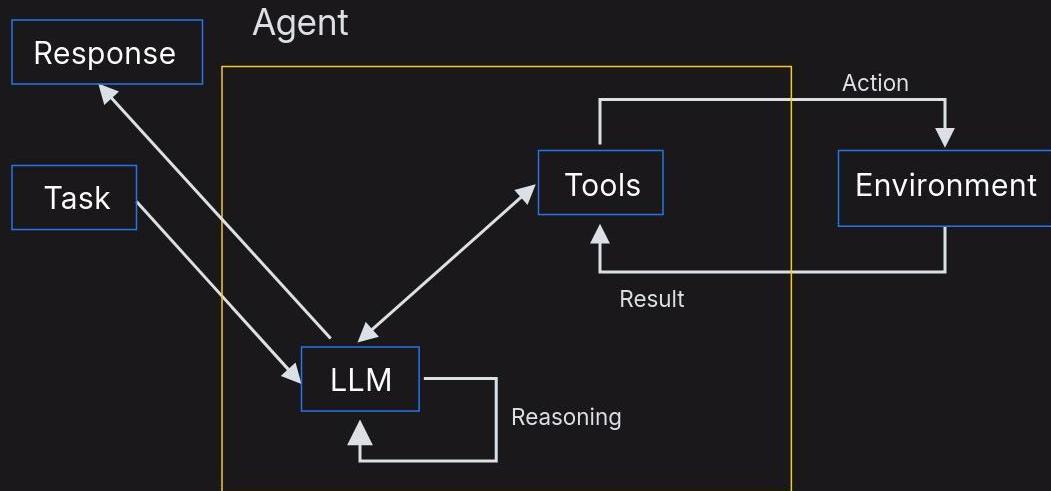
- **Human-in-the-Loop:**

Built-in persistence layer enhances human-agent interaction patterns.

- **Streaming First:**

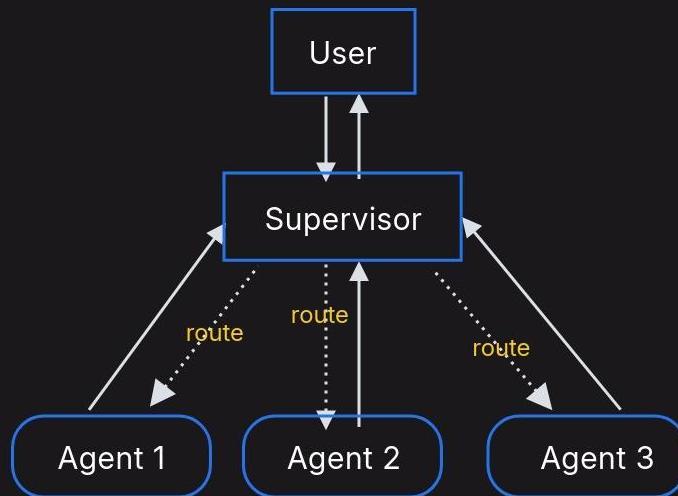
Supports streaming of events and tokens, providing real-time feedback to users.

# Standard Agentic AI System Workflow



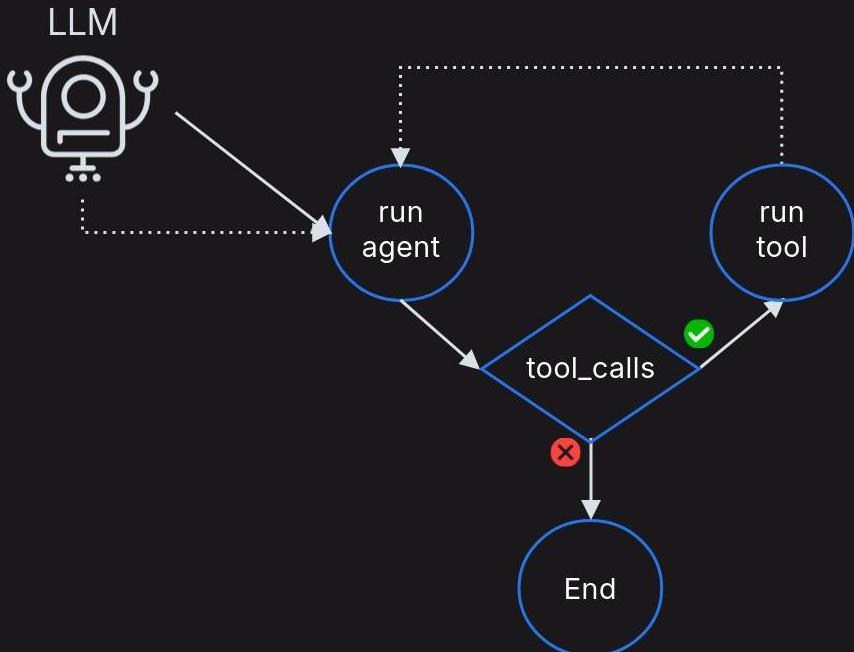
- Complex LLM applications often use cycles during execution.
- These cycles often use the LLM to reason about what to do next in the cycle, like Chain of Thought.
- This can essentially be thought of as running an LLM in a for-loop.
- These types of systems are commonly called AI agents.
- LangGraph is the perfect framework to model these agents as graphs, and LangChain recommends it.

# Multi-Agent AI System Workflow



- Multi-Agent workflows involve multiple independent agents powered by LLMs connected in a specific way
- Each agent can have its prompt, LLM, tools, and other custom code to collaborate with the other agents if needed
- LangGraph makes each agent a node in the graph. The control flow is managed by edges, and they communicate by adding to the graph's state

# LangGraph Agent Graph Components



- LangGraph treats Agent workflows as a cyclical Graph structure
- Main features
  - **Nodes:** Functions or LangChain Runnable objects such as tools.
  - **Edges:** Specify directional paths between nodes
  - **Stateful Graphs:** Manage and update state objects while processing data through nodes.
- LangGraph leverages this to facilitate cyclical LLM call executions with state persistence which is often required for AI Agents

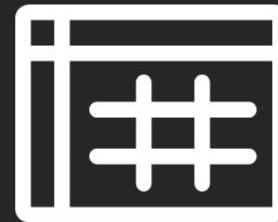
## LangGraph models agent workflows as **stateful** graphs

Stateful means all the steps executed are stored in their state to keep track of the overall execution flow and all variables, data, tool calls, and responses (e.g. as key : value pairs in a python dictionary).

# Key Components in LangGraph Agentic Systems

## States

- **States** represent the shared context or data passed and updated as the agent (graph) progresses through steps (nodes).
- Encapsulates all necessary information for the graph to function effectively, including:
  - Input data
  - Intermediate results
  - Execution metadata
- Can be any Python type, but typically a **TypedDict** or **Pydantic BaseModel**.

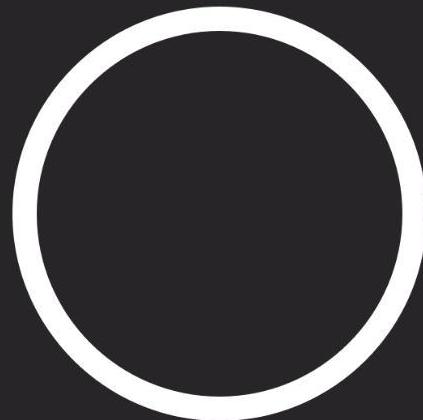


# Key Components in LangGraph Agentic Systems

## Nodes

---

- **Nodes** are Python functions that define the logic of your agent at different steps.
- They take the current state as input, process it, and return an updated state.
- Nodes can represent:
  - Individual functions or computations.
  - Tools the AI agent can use (e.g., calling an LLM).
  - API interactions or tool executions.



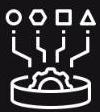
# Key Components in LangGraph Agentic Systems

## Edges

- **Edges** define the connections and routing logic between nodes.
- They determine the next node to execute based on the current state.
- Can represent:
  - **Conditional branches:** Dynamic paths based on state conditions.
  - **Fixed transitions:** Predefined execution paths.
- Enable flexible and dynamic execution flows.



# Key Components in LangGraph Agentic Systems



## State Graph & Schema:

This is typically used to define the overall state of the graph which includes the schema (variables and data types) which would be passed throughout the different nodes in the graph as the agent executes



## Reducers:

Often the same state variables (e.g messages) will be updated multiple times in different nodes.

Instead of overwriting the state variable, we use reducers to store all the messages (e.g appending all new updates into the same messages variable)

# Key Components in LangGraph Agentic Systems



## ToolNodes:

ToolNode is a LangChain Runnable that takes graph state (with a list of messages) as input and outputs state update with the result of tool calls.



## Normal Edges:

Goes directly from one node to the next, where each node is a function



## Conditional Edges:

Call a function to determine which node(s) to go to next.

# How Does LangGraph Build AI Agents?

```
●●●  
tavily_search = TavilySearchAPIWrapper()  
  
@tool  
def search_web(query: str) -> list:  
    """Search the web for a query."""  
    # Perform web search on the internet  
    results = tavily_search.raw_results(  
        query=query,  
        max_results=5,  
        include_raw_content=True  
    )  
    docs = results['results']  
    return docs
```

Tools

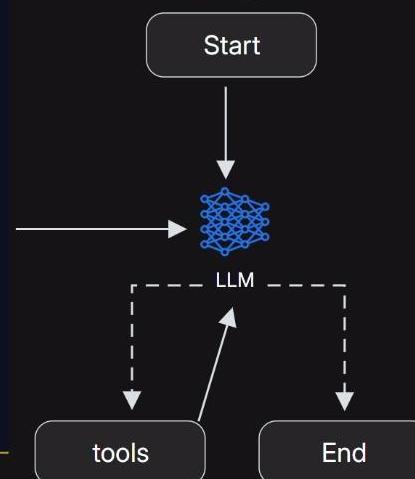
```
●●●  
# define Agent State  
class State(TypedDict):  
    messages: Annotated[list, add_messages]  
  
# bind tools to LLM  
tools = [web_search]  
llm = ChatOpenAI(model='gpt-4o')  
aug_llm = llm.bind_tools(tools)  
  
# create tool node function  
tool_node = ToolNode(tools=tools)  
  
# create LLM node function  
def llm_node(state: State) -> State:  
    SYS_PROMPT = '....' # system instructions  
    current_state = state["messages"]  
    state_with_prompt = SYS_PROMPT + current_state  
    response = [aug_llm.invoke(current_state)]  
    # update agent state  
    return {"messages": response}
```

Node Functions

Agent Graph

```
●●●  
# Build the graph  
graph = StateGraph(State)  
  
# Add nodes  
graph.add_node("llm", llm_node)  
graph.add_node("tools", tool_node)  
  
# Add edges  
graph.add_edge(START, "llm")  
# Conditional tool call or generate response  
graph.add_conditional_edges(  
    "tool_calling_llm",  
    tools_condition,  
    ["tools", END]  
)  
graph.add_edge("tools", "tool_calling_llm")  
  
# Compile Agent Graph  
agent = graph.compile()
```

ReAct Agent



# How Does LangGraph Execute AI Agents?

## Initial State

```
1
state: {
  messages: [('user', 'What are latest quantum
  developments?')]
}
next: Node LLM
checkpoint_id: abc123
thread_id: t001
```

## Tool Call Request

```
2
state: {
  messages: [('user', 'What are latest quantum
  developments?'),
  ('assistant', 'I'll search for quantum
  info.')]
}
Tool Request: web_search
Params: {"query": "quantum 2025"}
next: Tool Execution
checkpoint_id: def456
thread_id: t001
```

## Tool Response

```
3
state: {
  messages: [('user', 'What are latest quantum
  developments?'),
  ('assistant', 'I'll search for quantum
  info.'),
  ('tool', 'Found 10 articles...')]
}
Tool Response: search_results
Content: IBM quantum, Google advances...
next: Node LLM
checkpoint_id: ghi789
thread_id: t001
```

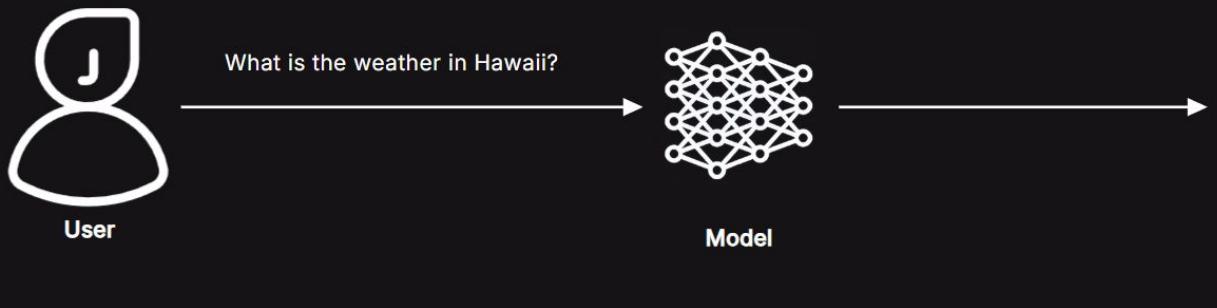
## LLM Processing

```
4
state: {
  messages: [('user', 'What are latest quantum
  developments?'),
  ('assistant', 'I'll search for quantum
  info.),
  ('tool', 'Found 10 articles...'),
  ('assistant', 'Key quantum developments...')]
}
next: Human Review
checkpoint_id: jkl012
thread_id: t001
```

## Final State

```
5
state: {
  messages: [('user', 'What are latest quantum
  developments?'),
  ('assistant', 'I'll search for quantum
  info.),
  ('tool', 'Found 10 articles...'),
  ('assistant', 'Key quantum developments...'),
  ('user', 'Tell me about IBM?')]
}
next: Node LLM
checkpoint_id: mno345
thread_id: t001
```

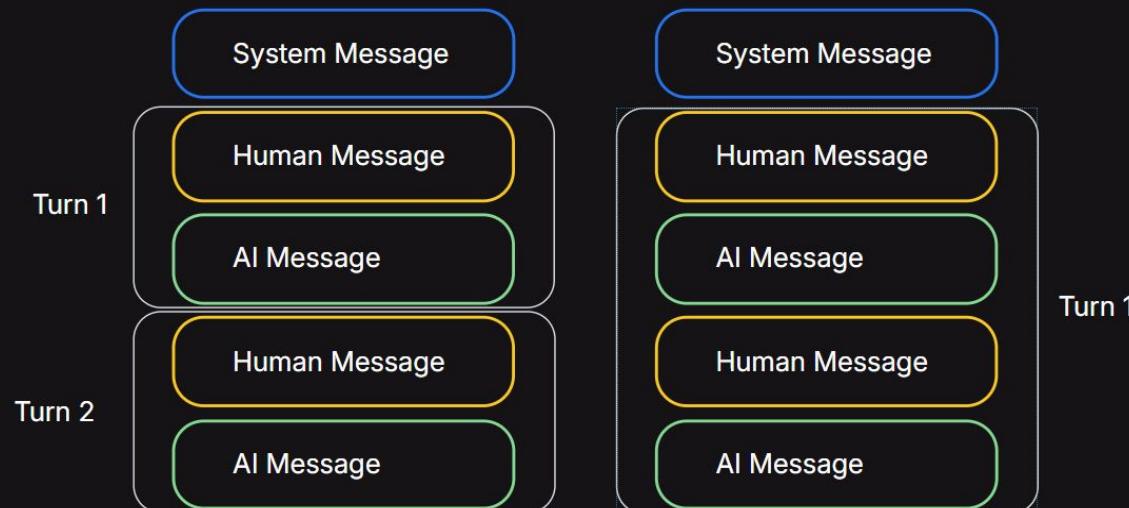
# Tools for Accessing the Outside World



```
AIMessage(  
    tool_calls=[{  
        name: "get_weather"  
        args: {  
            location: "Hawaii",  
        },  
        id: "call_abc123",  
        type: "tool_call"  
    }]  
)
```

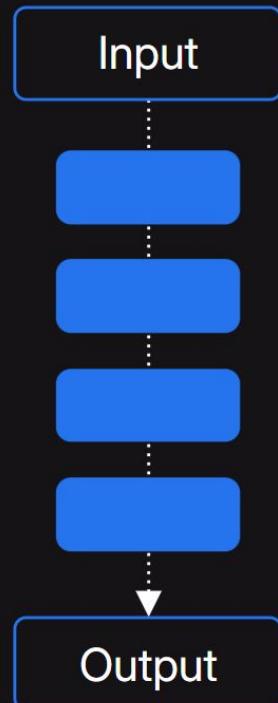
# Memory (short-term)

- Component necessary for applications with conversational interfaces.
- It allows access to a window of previous messages. We can think of it as short-term memory.
- Special caution must be taken with the size of this window to avoid exceeding the context limit of the LLM.



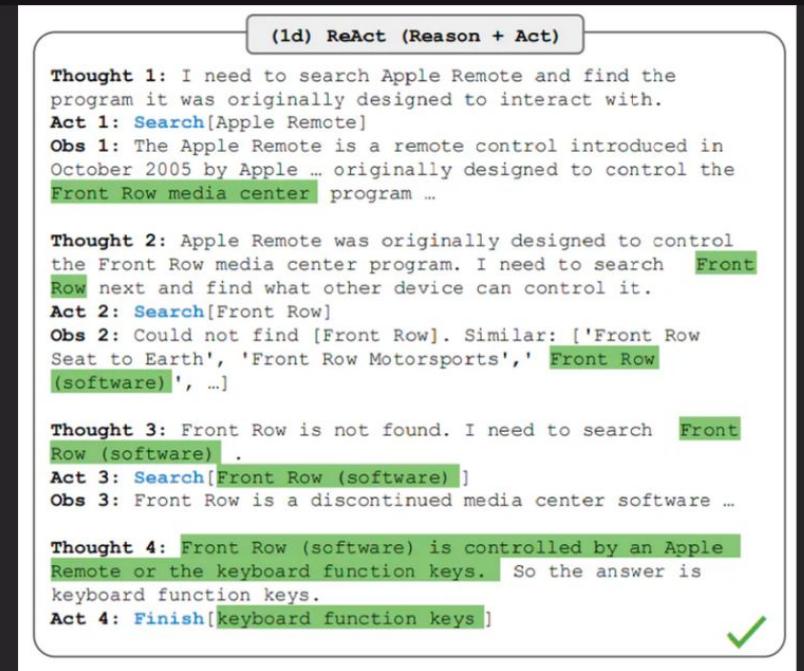
# Planning and Reasoning Heuristics

CoT - Chain Of Thought

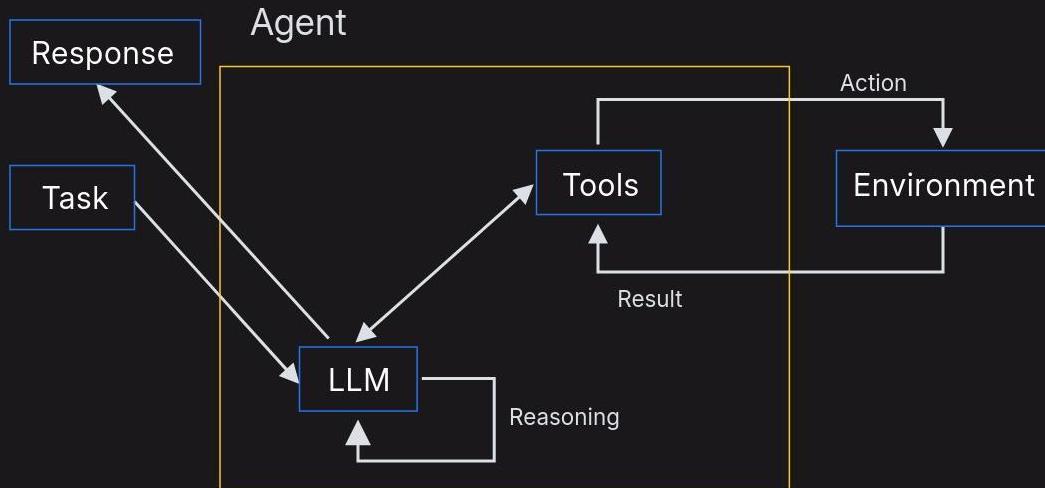


# Planning and Reasoning Heuristics - ReAct

- ReAct is the most famous Reasoning technique.
- Consists of a loop of Thought - Action - Observation
- Every framework, nowadays, uses ReAct under the hood.



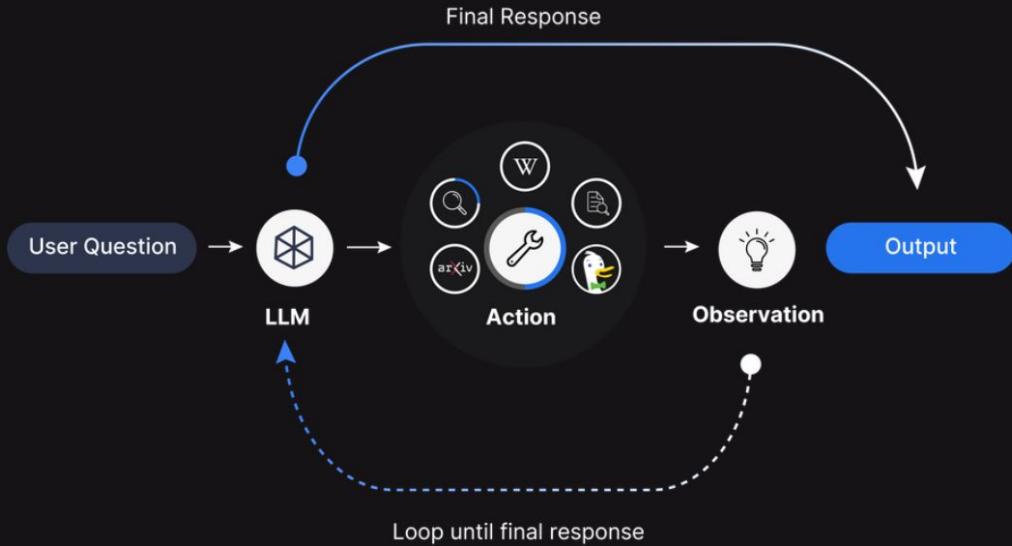
# ReAct Agentic AI Workflow



- This agentic workflow consists of a large language model (LLM) and tools
- The ReAct methodology is used to run the agent:
  - If the agent says to take an action (i.e. call tool), relevant tools are run and results are returned to the agent
  - If the agent did not ask to run tools, the response output is sent back to the user

# How do AI Agents work?

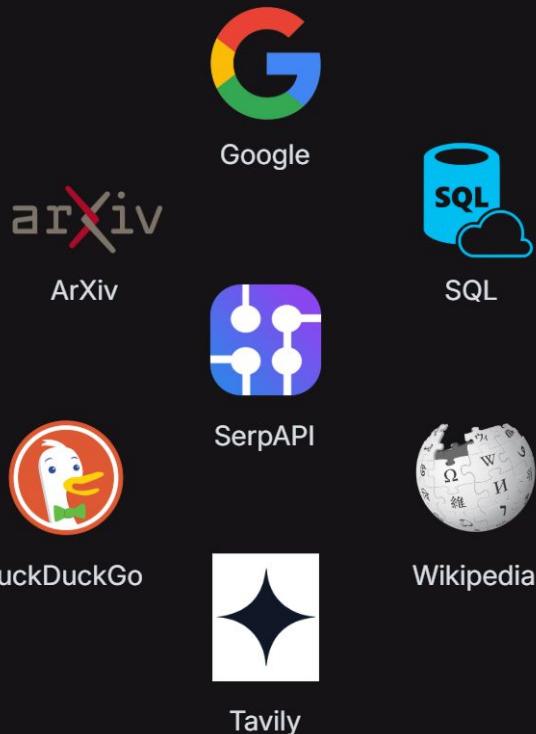
- Agents are systems that use an **LLM as a reasoning engine** to determine which actions to take and what the inputs to those actions should be.
- The results of those actions can then be fed back into the agent and it determines whether more actions are needed, or whether it is okay to stop.



# What are Tools?



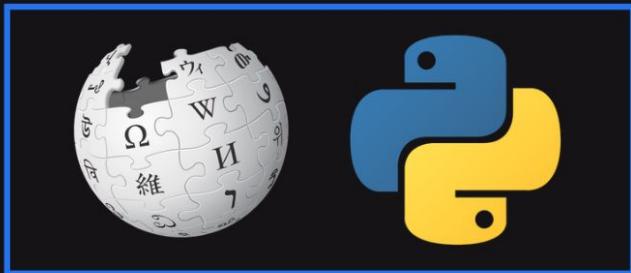
TOOLS



- Interfaces that an agent, a chain, or a chat model / LLM can use to interact with the world
- Given a list of available tools and a prompt, an LLM can request that one or more tools be invoked with appropriate arguments

# How do Tools Work?

Query: Microsoft



Wikipedia API



**Microsoft**

Article Talk

From Wikipedia, the free encyclopedia

**Microsoft Corporation** is an American multinational technology conglomerate headquartered in [Redmond, Washington](#).<sup>[2]</sup> Founded in 1975, the company became highly influential in the [rise of personal computers](#) through software like [Windows](#), and the company has since expanded to [Internet services](#), [cloud computing](#), [video gaming](#) and other fields. Microsoft is the [largest software maker](#), one of the [most valuable public U.S. companies](#),<sup>[a]</sup> and one of the [most valuable brands](#) globally.

Microsoft was founded by [Bill Gates](#) and [Paul Allen](#) to develop and sell [BASIC interpreters](#) for the [Altair 8800](#). It rose to dominate the personal computer operating system market with [MS-DOS](#) in the mid-1980s, followed by [Windows](#). The company's 1986 [initial public offering \(IPO\)](#) and subsequent rise in its share price created three

# How do Tools Work?

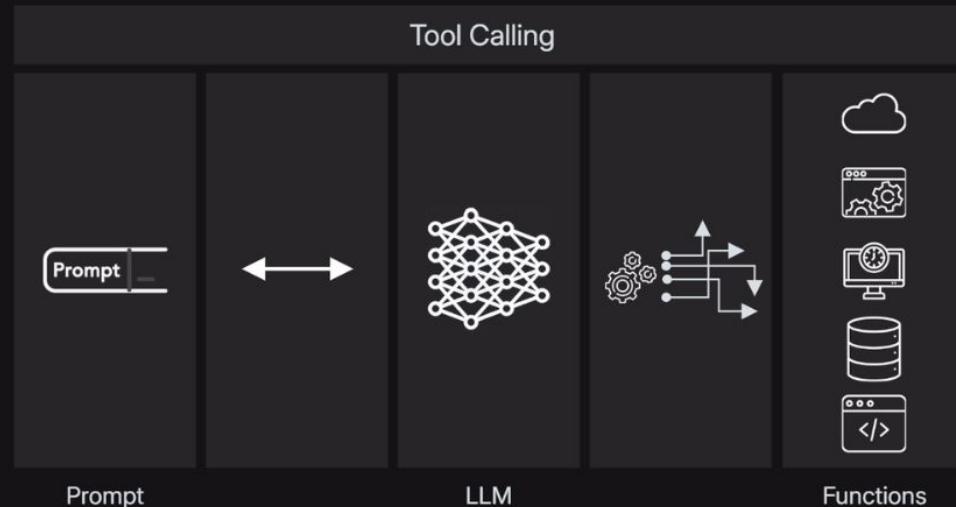
Query: How is the weather in Zurich?



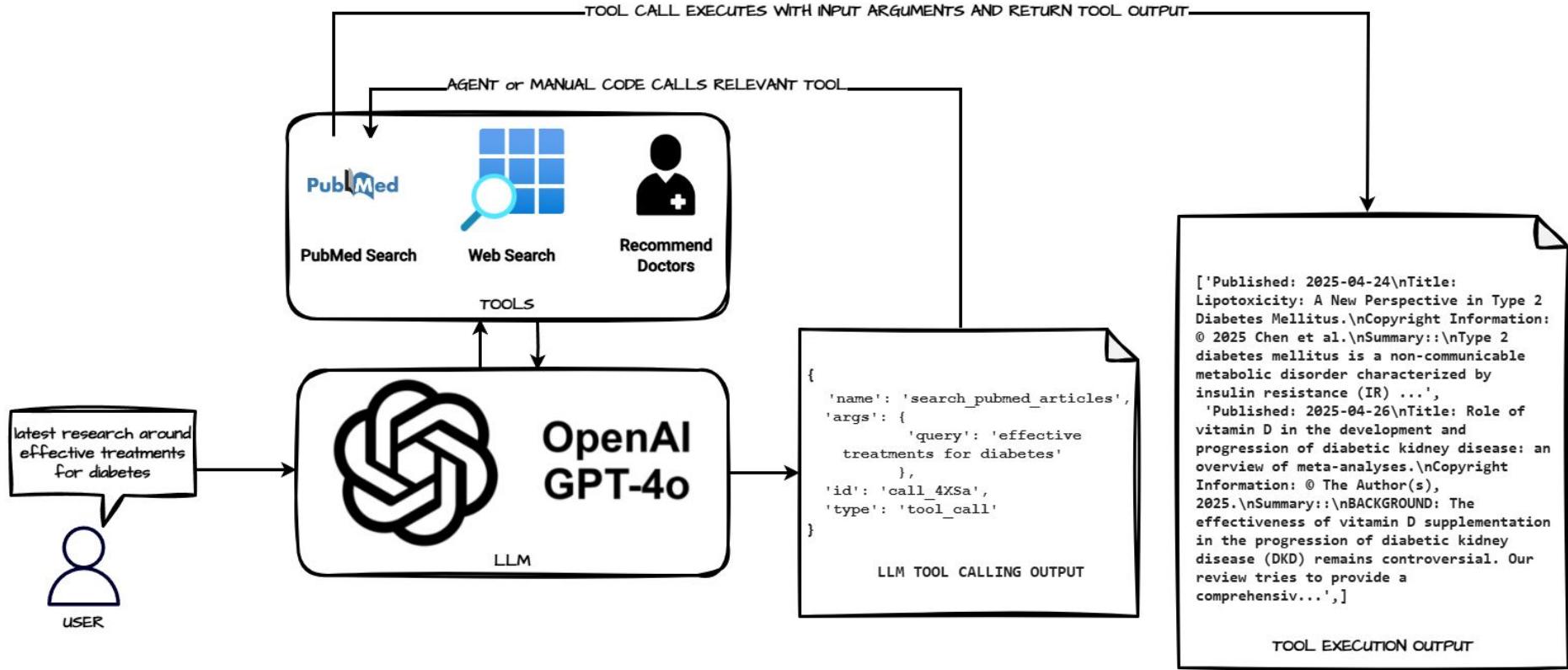
```
{
  "location": {
    "name": "Zurich",
    "region": "",
    "country": "Switzerland",
    "lat": 47.3667,
    "lon": 8.55,
    "tz_id": "Europe/Zurich",
    "localtime_epoch": 1736620060,
    "localtime": "2025-01-11 19:27"
  },
  "current": {
    "last_updated_epoch": 1736619300,
    "last_updated": "2025-01-11 19:15",
    "temp_c": -1.6,
    "temp_f": 29.1,
    "is_day": 0,
    "condition": {
      "text": "Clear",
      "icon": "//cdn.weatherapi.com/weather/64x64/night/113.png",
      "code": 1000
    },
    "wind_mph": 2.2,
    "wind_kph": 3.6,
    "wind_degree": 52,
    "wind_dir": "NE",
    "pressure_mb": 1027.0,
    "pressure_in": 30.33,
    "precip_mm": 0.0,
    "precip_in": 0.0,
    "humidity": 80,
    "cloud": 0
  }
}
```

# What is Tool or Function Calling

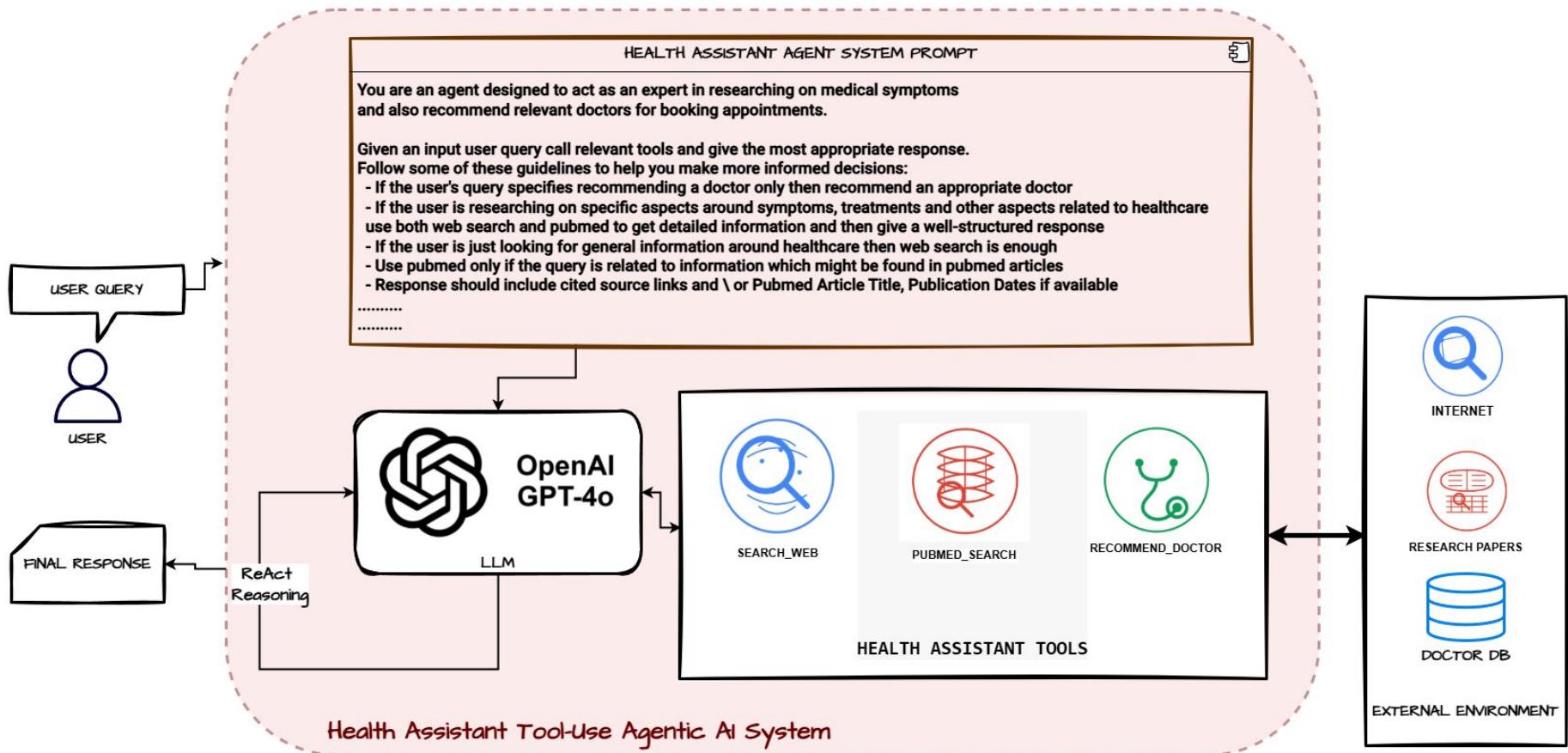
- Function calling enables Large Language Models (LLMs) to interact with the external environment, rather than just generating text responses.
- LLMs can identify the tools or functions needed for additional information to formulate an appropriate response.
- Access to pre-defined tools like web search is required for function calling, where the LLM indicates the specific tool and arguments needed.
- Agents must independently call the tools as function calling does not automate this process; it simply specifies which tools are needed.



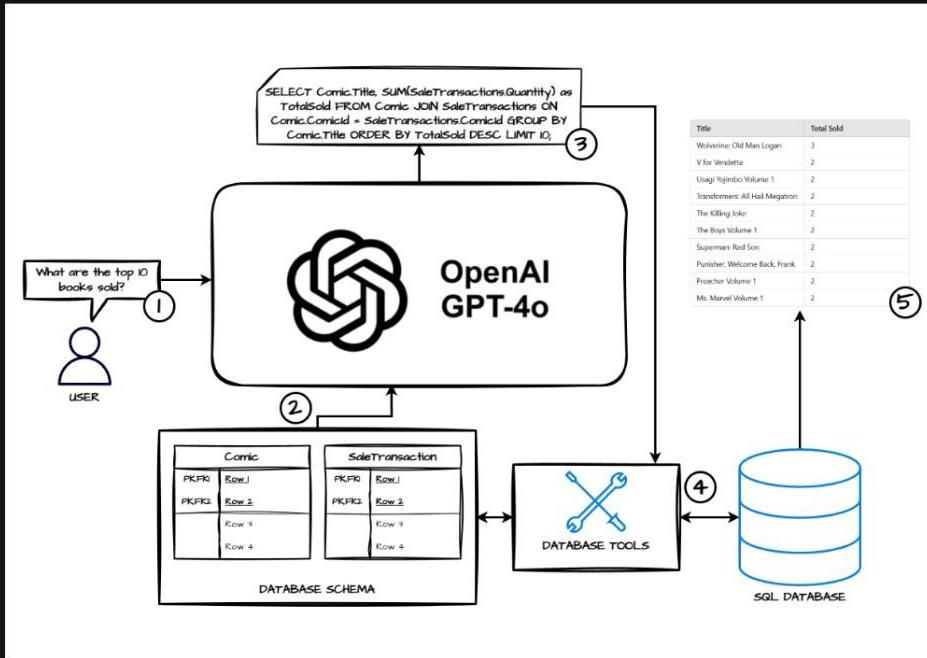
# Tool Calling Workflow



# ReAct Agent Architecture

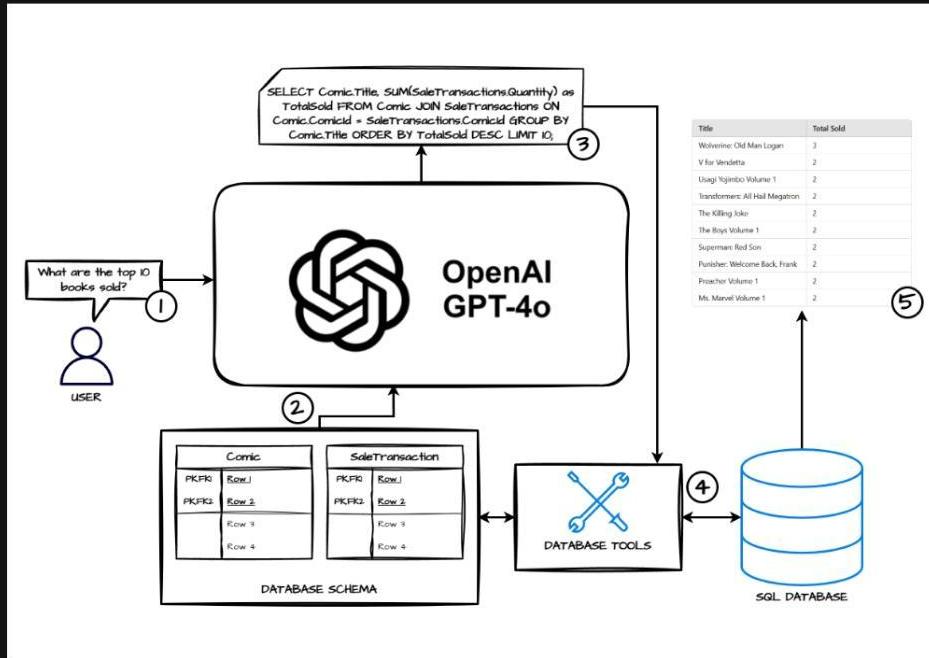


# What is Text2SQL?



- A Text-to-SQL system is an AI system that translates natural language questions into SQL queries to retrieve information from a database.
- For example, you can ask, "What are the top 10 books sold?", and the system generates the corresponding SQL query automatically and then executes it to fetch the result.

# What is Text2SQL?



## How It Works:

- **Input**
  - A natural language question
- **Processing**
  - The system parses the input and maps it to database tables, columns, and relationships.  
(In Generative AI systems LLMs do this.)
- **Output**
  - An SQL query is generated
- **Execution**
  - The SQL query is run on the database to retrieve the results.

# Standard Text2SQL System Components & Flow

## User Question

What are the Top 10 most popular comics?

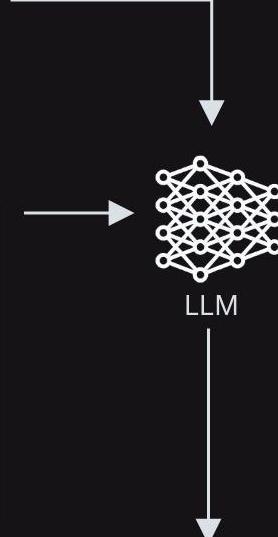
## Schema

TABLE SaleTransactions

  TABLE Comic

    TABLE Comic details:

```
"ComicId" INTEGER  
"Title" NVARCHAR  
"PublisherId" INTEGER  
"Genre" NVARCHAR  
"Price" NUMERIC(10, 2)  
"ReleaseDate" DATETIME,  
PRIMARY KEY ("ComicId"),  
FOREIGN KEY("PublisherId")
```



## Generated SQL Query:

```
SELECT Comic.Title, SUM(SaleTransactions.Quantity) as TotalSold FROM Comic JOIN  
SaleTransactions ON Comic.ComicId = SaleTransactions.ComicId GROUP BY Comic.Title  
ORDER BY TotalSold DESC LIMIT 10;
```

## Execution Results

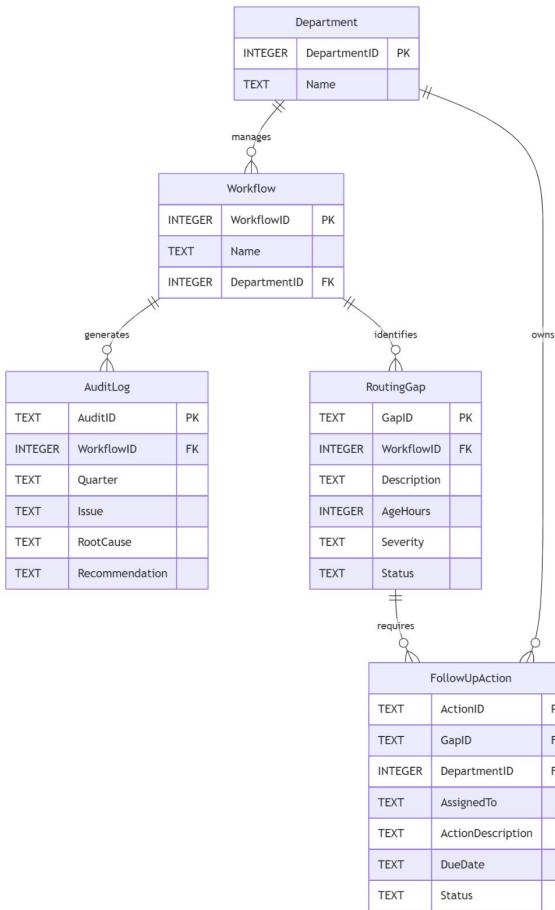
Title	Count
Wolverine: Old Man Logan	3
V for Vendetta	2
Usagi Yojimbo Volume 1	2
Transformers: All Hail Megatron	2
The Killing Joke	2
The Boys Volume 1	2
Superman: Red Son	2
Punisher: Welcome Back, Frank	2
Preacher Volume 1	2
Ms. Marvel Volume 1	2

# Text2SQL in the Age of Generative AI & Agentic Systems



- With the advent of powerful fine-tuned LLMs like GPT-4o, Llama 3.3 and more, we do not need to use hard-coded rules
- The idea is to formulate your query as a prompt, add in necessary details like database table schemas and relationships, and ask the LLM to generate the SQL query
- Agentic Flows can help with multi-step reasoning, checks, retrieval of table schemas, and executing SQL queries to get results
- We will implement this in future videos*

# Our Database Schema



## Our Database Schema – OperationsInsights.db

- Department Management:** Maintains business unit information such as Finance, Procurement, Claims, etc. Each department owns one or more workflows.
- Workflow Mapping:** Captures operational processes like Invoice Processing or Vendor Onboarding, and links them to their responsible departments.
- Audit Logging:** Stores historical audit findings including issues, root causes, and recommended improvements. Useful for analyzing SLA breaches over time.
- Routing Gap Tracking:** Identifies unresolved process exceptions in real-time. Includes age (in hours), severity, and open/closed status for each workflow gap.
- Follow-Up Action Management:** Tracks tasks created to resolve routing gaps. Each action is assigned to a department or team with a deadline and current status.

# Our Database Schema

## 1. Department

### Schema

Field	Type	Description
DepartmentID	INTEGER	Unique ID for each department
Name	TEXT	Department name (e.g., Finance Ops)

- Used to categorize workflows and assign actions.

### Sample Rows

DepartmentID	Name
1	Finance Operations
2	Procurement & Vendor Mgmt
3	Claims Processing Unit

# Our Database Schema

## 2. Workflow

### Schema

Field	Type	Description
WorkflowID	INTEGER	Unique ID for each workflow
Name	TEXT	Workflow name (e.g., Invoice Processing)
DepartmentID	INTEGER	Linked department

- ◆ Represents specific business processes owned by departments.

### Sample Rows

WorkflowID	Name	DepartmentID
101	Invoice Processing	1
102	Vendor Onboarding	2
103	Claims Approval	3

# Our Database Schema

## 3. AuditLog



### Schema

Field	Type	Description
AuditID	TEXT	Unique audit identifier
WorkflowID	INTEGER	Linked workflow
Quarter	TEXT	Audit period (e.g., Q1-2024)
Issue	TEXT	What went wrong
RootCause	TEXT	Diagnosed root cause
Recommendation	TEXT	Suggested fix

- ◆ Captures historical SLA failures and improvement recommendations.

### Sample Rows

AuditID	WorkflowID	Quarter	Issue	RootCause	Recommendation
A001	101	Q1-2024	SLA breach on approvals	Static approval chain	Add fallback approvers
A002	102	Q1-2024	Vendor doc intake failed	No validation check	Enforce schema validation at intake

# Our Database Schema

## 4. RoutingGap

### Schema

Field	Type	Description
GapID	TEXT	Unique ID for routing gap
WorkflowID	INTEGER	Related workflow
Description	TEXT	What the gap is
AgeHours	INTEGER	Time issue has remained unresolved (in hours)
Severity	TEXT	Severity level (e.g., High, Medium, Low)
Status	TEXT	Whether the issue is Open or Resolved

- ◆ Tracks open operational issues with delay age and severity.

### Sample Rows

GapID	WorkflowID	Description	AgeHours	Severity	Status
RG001	102	Unmonitored exception queue	72	High	Open
RG002	101	Inactive approver in routing chain	48	Medium	Open

# Our Database Schema

## 5. FollowUpAction

### Schema

Field	Type	Description
ActionID	TEXT	Unique action ID
GapID	TEXT	Linked routing gap
DepartmentID	INTEGER	Department assigned to fix the issue
AssignedTo	TEXT	Assignee email or team name
ActionDescription	TEXT	What action is to be taken
DueDate	TEXT	Action deadline
Status	TEXT	Pending / Done / Overdue

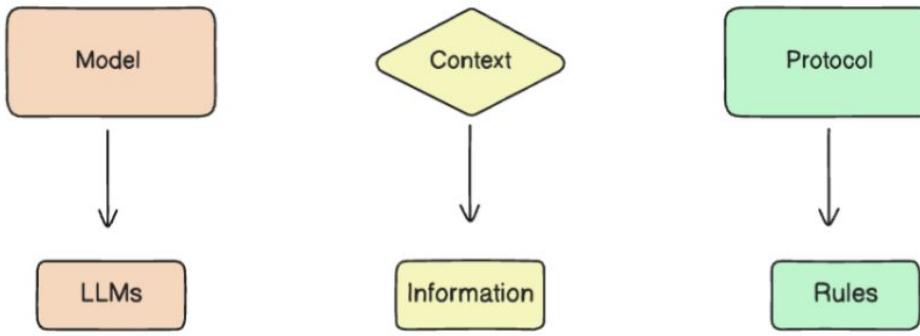
- Logs resolution tasks for routing issues, with owners and deadlines.

### Sample Rows

ActionID	GapID	DepartmentID	AssignedTo	ActionDescription	DueDate	Status
ACT001	RG001	2	ops@company.com	Configure monitored inbox for exceptions	2024-07-25	Pending
ACT002	RG002	1	sla_team@company.com	Add fallback approver	2024-07-28	Pending

# What is Model Context Protocol?

It is a powerful open standard that was launched by Claude's parent company, Anthropic, in November 2024.



**Model:** The LLM (e.g., Claude, GPT-4) generates responses.

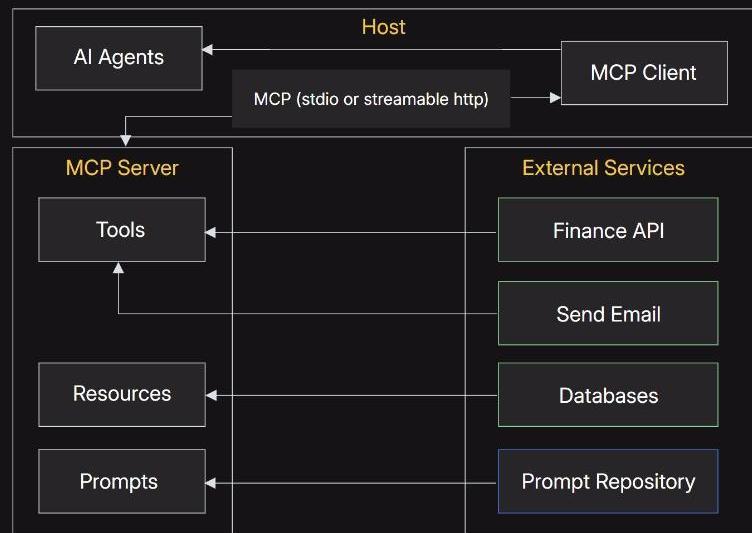
**Context:** Additional input (documents, PDFs, prompts, databases) for meaningful replies.

**Protocol:** Rules enabling the model to access and use structured context.

# What is MCP?

Model Context Protocol (MCP) is an open standard introduced by Anthropic to standardize how AI applications (chatbots, IDE assistants, or custom agents) connect with external tools, data, and prompts

- Think of MCP as the USB-C for AI integrations:  
M (applications) and N (tools) do not need an MxN integration, with MCP it is now a M + N problem
- MCP defines a client-server architecture where:
  - Host – Your local system or servers where apps, agents, and chatbots run
  - MCP Client – Runs along with your apps and agents on your host
  - MCP Server – Exposes Tools, Resources, and Prompts to the host client using the model context protocol (MCP)
- MCP goes beyond just standardizing tool-calling, use it for:
  - **Access tools** to empower agent actions
  - **Access data** from various sources
  - **Access prompts** and instructions for specific tasks



# Do we really need MCP?

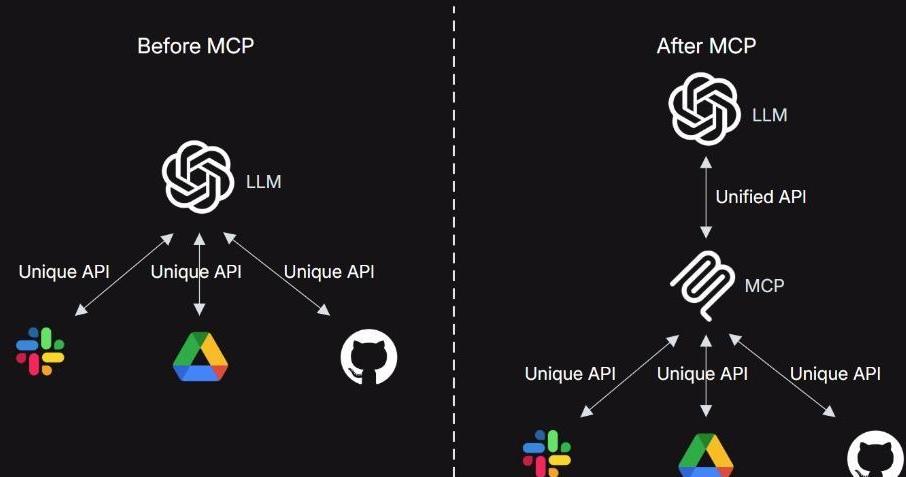
MCP has been hyped up a lot over the last few months, is it worth the hype?

- Direct Consumers

- Super useful as every major data or service provider now provides ready-made MCP servers for you to connect your apps and start using their services and tools

- Enterprise Users & Builders

- It depends on how you use it:
  - ✗ If you are building your own tools & data APIs to connect with your apps and agents, MCP is pointless
  - ✓ If you want to expose your tools & APIs to other departments and users, then it is useful
  - ✗ Use 3rd-party MCP servers with caution, considering security and regulatory implications



# What is Financial Analysis?

Financial analysis is the process of examining a company's financial data to understand its performance and make decisions (like buying or selling the stock).



## Key Focus Areas

---

- Looking at **income** (money coming in).
- Checking **expenses** (money going out).
- Measuring **profitability** (how much money is left after costs).
- Assessing **assets** (what a company owns) and liabilities (what it owes).
- Looking at current **news** (of how the company is faring)

# What is Financial Analysis?

Financial analysis is the process of examining a company's financial data to understand its performance and make decisions (like buying or selling the stock).



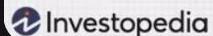
## Common Tools

---

- Company stock price data and trends
- Fundamental indicators, like profit margin, price-to-earnings ratio, debt-to-equity ratio
- Recent company events, news, earnings call reports

# Key Concepts - Stock Price Indicators

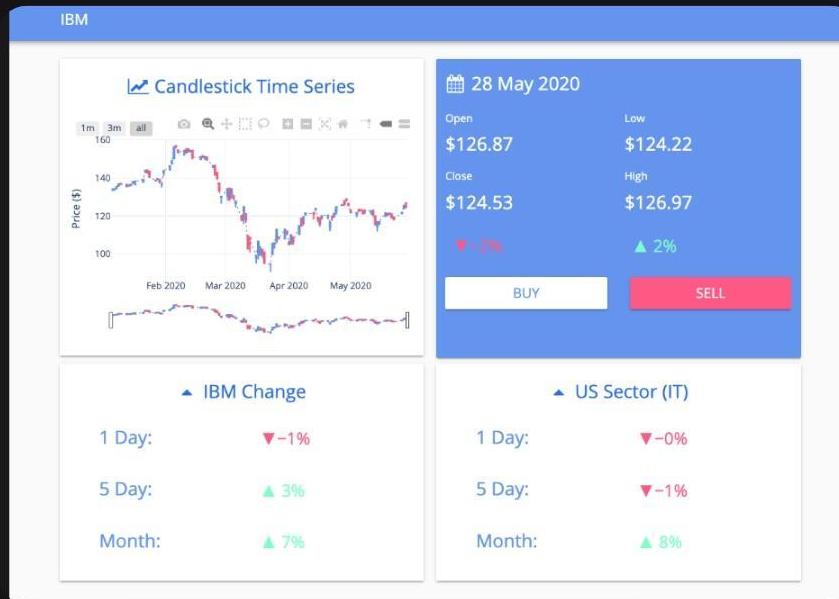
Microsoft Corp \$46.41		+0.37
Last trade	46.41	
Trade time	Sep 26 04:00 PM Eastern Daylight Time	
Change	0.37 (0.80%)	
Prev close	46.04	
Open	45.93	
Bid	46.39	
Ask	46.40	



## Stock Price Quote

- Purpose
  - Fetches the latest stock price information
- Data Retrieved
  - Includes details like the current price, previous close (`prev_close`), change in price (`change`), percentage change (`change_percent`), 52-week high (`year_high`), and 52-week low (`year_low`)
- Core Concept
  - Provides a snapshot of the stock's current market status, including recent performance metrics and key benchmarks

# Key Concepts - Stock Price Indicators



## Stock Price Performance

- Purpose

- Calculates the stock's price performance over various time periods

- Data Retrieved

- Includes metrics such as one-day, one-week, one-month, and year-to-date (`ytd`) performance, as well as six-month and one-year changes. It may also provide weekly and monthly volatility (`volatility_week`, `volatility_month`), trading volume (`volume`), average volume, and relative volume (`relative_volume`)

- Core Concept

- Offers insights into how the stock's price has changed over different time frames, along with additional metrics like trading activity and market sentiment indicators (e.g., `analyst_score`)

# Key Concepts - Stock Price Indicators

Stock Price Close and Volume by Date



## Stock Price Historical Data

- **Purpose**
  - Retrieves historical price data for the stock
- **Data Retrieved**
  - Includes daily details such as the date, opening price (`open`), highest price (`high`), lowest price (`low`), closing price (`close`), and trading volume (`volume`)
- **Core Concept**
  - Provides a detailed view of the stock's performance over time, showing trends in price movements and trading activity that can help with historical analysis and forecasting.

# Key Concepts - Stock Fundamental Analysis



**Fundamental Analysis**  
[fən-də-ˈmen-təl ə-ˈna-lə-səs]  
A method of determining a stock's real or "fair market" value.

Investopedia

## What is Stock Fundamental Analysis?

- **Definition**
  - Fundamental analysis involves evaluating a stock's intrinsic value by analyzing financial statements, economic conditions, and company performance.
- **Purpose**
  - It helps investors determine whether a stock is overvalued or undervalued, guiding buy, hold, or sell decisions.
- **Key Focus**
  - Examines revenue, earnings, profit margins, and growth potential alongside broader industry and market conditions.
- **Goal**
  - To identify long-term investment opportunities by understanding a company's financial health and future prospects.

# Key Concepts - Stock Fundamental Indicators & Metrics

Price-to-Earnings Ratio (P/E Ratio)

Price-to-Sales Ratio (P/S Ratio)

Debt-to-Equity Ratio

Return on Equity (ROE)

## Fundamental Ratios

- Financial Health Assessment
  - Provides key ratios like the Current Ratio, Quick Ratio, and Cash Conversion Cycle to evaluate a company's liquidity, operational efficiency, and short-term financial health.
- Profitability and Performance
  - Offers metrics such as Gross Profit Margin, Operating Profit Margin, and Return on Equity to analyze the company's profitability and how well it utilizes resources.

# Key Concepts - Financial News & General Market Indicators



## Importance of Financial News

- **Timely Information**

- Financial news provides real-time updates on company earnings, mergers, acquisitions, and other significant events that can influence stock prices. Being aware of these developments allows investors to react promptly to market changes.

# Key Concepts - Financial News & General Market Indicators



## General Market Insights

- **Market Activity Overview**
  - Tracks the most actively traded stocks by volume, helping investors identify where market interest is concentrated. It also highlights the top gainers (stocks with the highest price increases) and top losers (stocks with the largest price declines), offering insights into potential opportunities and market sentiment shifts.

# Financial Data Platforms

The screenshot displays the OpenBB Terminal Pro application window. It features several tabs and panels:

- Earnings Transcript:** Shows a transcript from NVIDIA's Second Quarter Earnings Call. The operator, Abby, welcomes everyone and introduces the speakers, Jensen Huang and Colette.
- OpenBB Copilot:** A tool for generating earnings reports. It shows the context "Earnings Questions" and "Earnings Transcripts NVDA". A purple bar at the top says "Answer all the earnings questions". Below it, "Step-by-step reasoning" is listed, followed by "Earnings Questions Answered".
  - Question 1: "Using the earnings transcript, create a table with columns: financial metric, value, sentence in the earnings where it was extracted from. Double check whether the information you are using is correct." The answer provides a table with columns "Financial Metric", "Value", and "Sentence in Earnings Transcript".
  - Question 2: "What is the guidance for the stock?" The answer states: "The guidance for NVIDIA stock for the third quarter of fiscal 2025 includes:
    - Total Revenue: Expected to be \$32.5 billion, plus or minus 5% sequentially.
    - EPS: Guidance range is \$1.10 to \$1.15 per share."
- Charting:** A chart showing NVIDIA Corp. ID - NASDAQ: NVDA over time, with various metrics like Price, Volume, and MACD.
- Share Statistics:** Displays financial metrics for NVDA:
  - Free Float Shares: 1.66 M
  - Shares Outstanding: 1.66 M
  - Free Float Percentage: 99.89%
- Watchlist:** A list of stocks with their last price, day change, and week change.

Name	Last	Day Change	Week Change
AMZN	145.80	+2.25%	+0.55%
NVDA	604.33	+2.37%	+0.46%

OpenBB a popular open-source Financial Platform

Financial data platforms provide investors and professionals with real-time market data, historical trends, financial statements, and analytical tools, enabling informed decisions with comprehensive market insights.

# Financial Data Platforms

The screenshot shows the OpenBB Terminal Pro application interface. On the left, there's a sidebar with navigation links like 'Data Connectors', 'Templates', 'MY DASHBOARDS', and 'SHARED WITH ME'. The main area has several windows open:

- Earnings Transcript (NVDA, 2025 Q1-Q2-Q3-Q4):** Displays a transcript from NVIDIA's Second Quarter Earnings Call.
- OpenBB Copilot (2023-06-01 Earnings Report Analysis):** A AI-powered tool for analyzing earnings reports. It shows a section titled 'Earnings Questions Answered' with a question about creating a table from the transcript.
- Charting (NVIDIA Corp. ID - NASDAQ):** A chart showing NVIDIA's stock price performance over time.
- Share Statistics (NVDA):** Provides key financial metrics: Free Float Shares (1.66 M), Shares Outstanding (1.66 M), and Free Float Percentage (99.89%).
- Watchlist (Edit Ticker):** A list of stocks including AMZN and NVDA, showing their last price, day change, and week change.

OpenBB a popular open-source Financial Platform

## Key Features of Financial Data Platforms:

- Real-Time Market Data**
  - Access to up-to-the-minute information on stock prices, trading volumes, and market indices.
- Historical Data**
  - Availability of past financial data for trend analysis and backtesting investment strategies.
- Financial Statements**
  - Detailed reports such as balance sheets, income statements, and cash flow statements for company analysis.
- Analytical Tools**
  - Features like charting, technical analysis indicators, and financial modeling capabilities.
- News and Research**
  - Integration of financial news, analyst reports, and market research to stay informed about market developments.

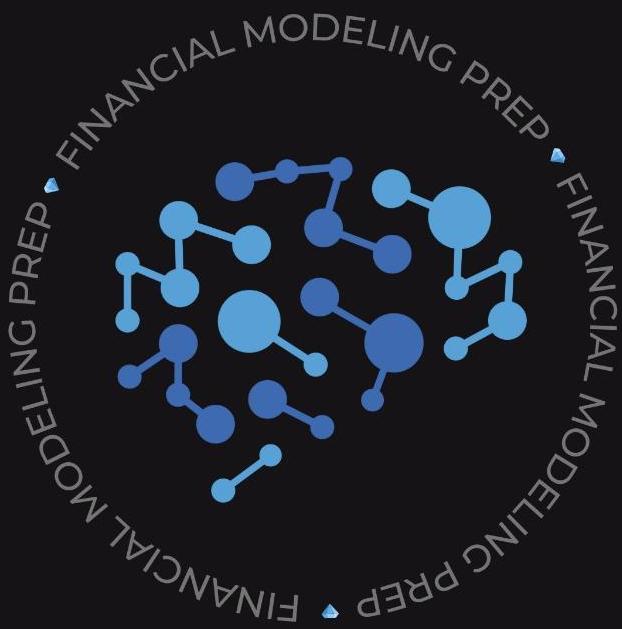
# Popular Financial Data Platforms & Providers



## OpenBB

An open-source financial analysis platform that integrates with numerous data sources, providing tools for equity, options, crypto, forex, and more.

# Popular Financial Data Platforms & Providers



## Financial Modeling Prep (FMP):

Offers a comprehensive suite of financial data, including historical prices, fundamental data, and financial ratios, catering to developers, traders, and analysts.

# Popular Financial Data Platforms & Providers



## yfinance

A Python library that enables users to access financial data from Yahoo Finance, including historical market data, financial statements, and stock information.

# Popular Financial Data Platforms & Providers



## Finviz

A financial visualization platform that provides advanced stock screeners, market maps, and analysis tools for traders and investors.

# Popular Financial Data Platforms & Providers



**Cboe**

One of the world's largest exchange holding companies, offering trading and investment solutions, including real-time market data and analytics for equities, options, and futures.

# Popular Financial Data Platforms & Providers



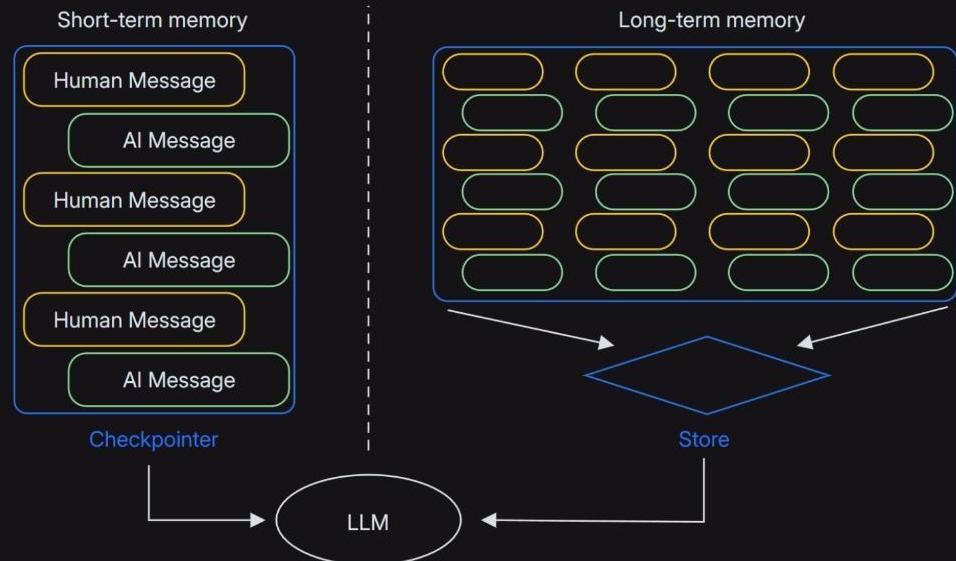
**TMX**

Operates the Toronto Stock Exchange and provides comprehensive market data, analytics, and trading solutions for Canadian and international markets.

# Memory Management for Agentic AI Systems

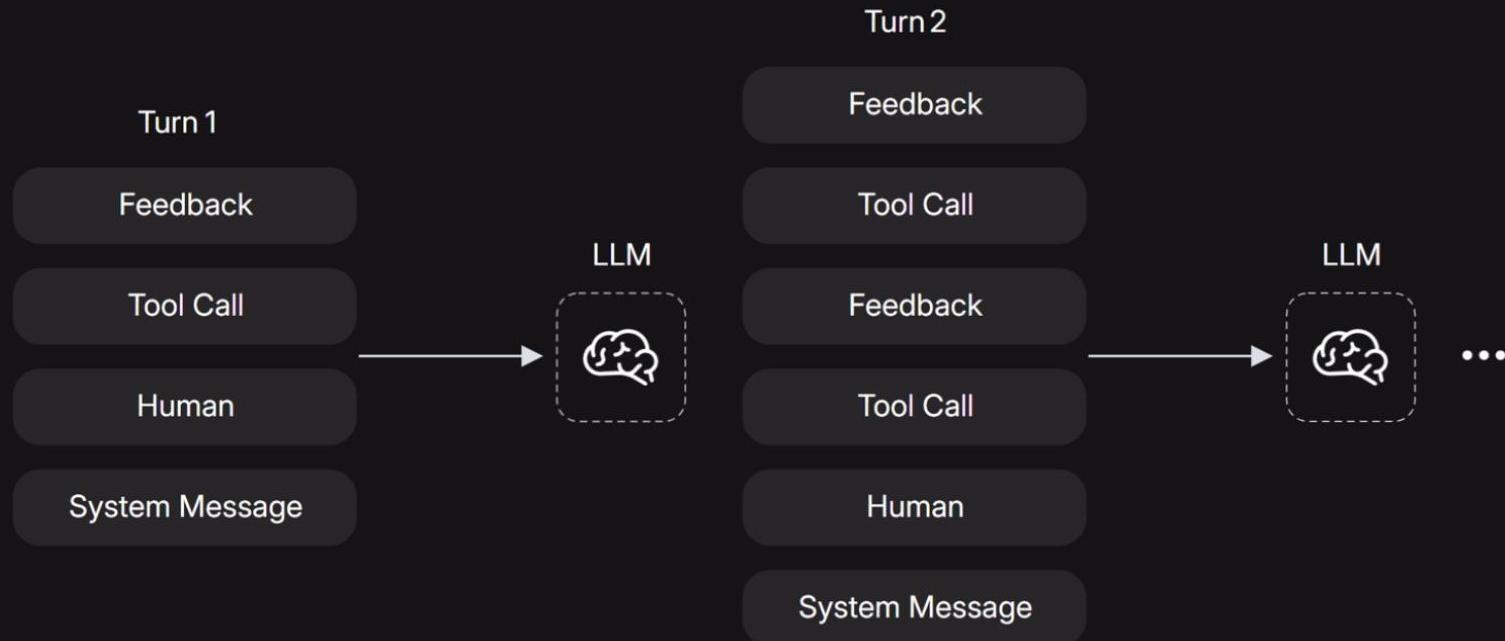
Memory enables Agentic AI Systems to have better interactions by leveraging past conversations, data, tool call results, and personalized preferences.

- Memory is a misnomer - Agents or LLMs can't really remember like humans!
- It is context engineering, to stuff past information into the context window of the Agent's LLM for future interactions
- Two major types of memory:
  - **Long-Term:** Storing all interaction information and transforming into useful artifacts like user preferences
  - **Short-Term:** Storing recent interaction information like conversations, tool calls, and more



# Why care about Memory Management for Agents?

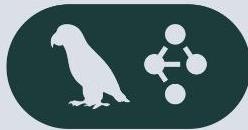
Agent context keeps increasing due to storing past interactions in memory. This can easily lead to the LLM hitting the dreaded 'Max Context Window Limit Exceeded' error!



# Popular Tools & Frameworks for Building Agentic AI Systems



memØ



## LangGraph

langchain-ai/  
langmem



12  
Contributors

258  
Used by

12  
Discussions

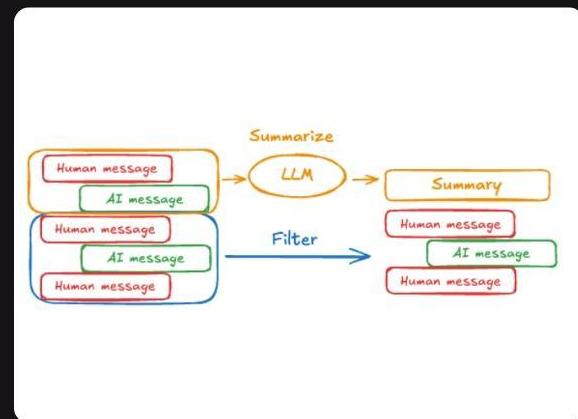
951  
Stars

110  
Forks

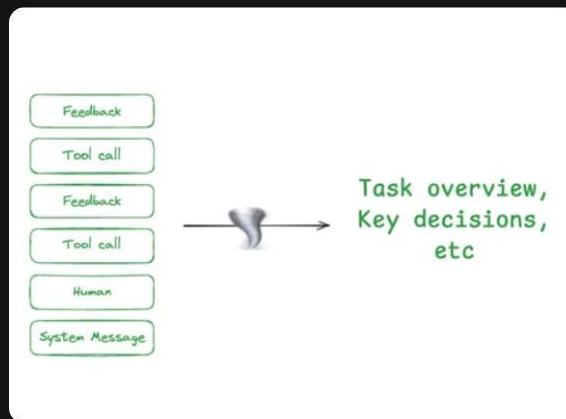


# Short-Term Memory Management Methodologies

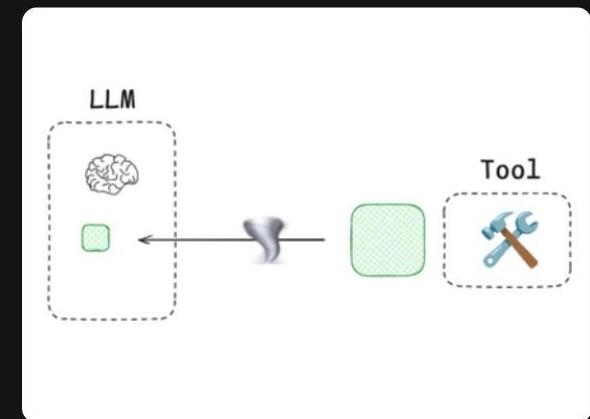
Short-term memory can be optimized by summarizing past history as well as tool responses. We can also filter out older historical messages.



Summarizing and\or Filtering out conversations



Summarizing History



Summarizing Tool Responses

# Real-World Example

## Long-Term Memory Management for Personalized Agents

LangMem has tools to transform long-term agent memory interactions into well-defined user-profiles or records to enable personalized experiences

```
from langmem import create_memory_manager
from pydantic import BaseModel

# define schema to store user preferences
class UserProfile(BaseModel):
    """Save the user's preferences."""
    name: str
    preferred_name: str
    response_style_preference: str
    special_skills: list[str]
    other_preferences: list[str]

manager = create_memory_manager(
    "openai:gpt-5",
    schemas=[UserProfile],
    instructions="Extract user preferences and settings",
    enable_inserts=False,
)

# Extract user preferences from agent conversations
# e.g conversation = # get interactions from the agent
conversation = [
    {"role": "user", "content": "Hi! I'm Alex but please call me Lex. I'm a wizard at Python and love making AI systems that don't sound like boring corporate robots 😑"}, {"role": "assistant", "content": "Nice to meet you, Lex! Love the anti-corporate-robot stance. How would you like me to communicate with you?"}, {"role": "user", "content": "Keep it casual and witty - and maybe throw in some relevant emojis when it feels right ✌️ Also, besides AI, I do competitive speedcubing!"}, {"role": "assistant", "content": "Great! Let's keep it casual and witty. I'm a big fan of competitive speedcubing myself. What's your favorite cube size?"}
]

# create user profile
profile = manager.invoke({"messages": conversation})[0]
```



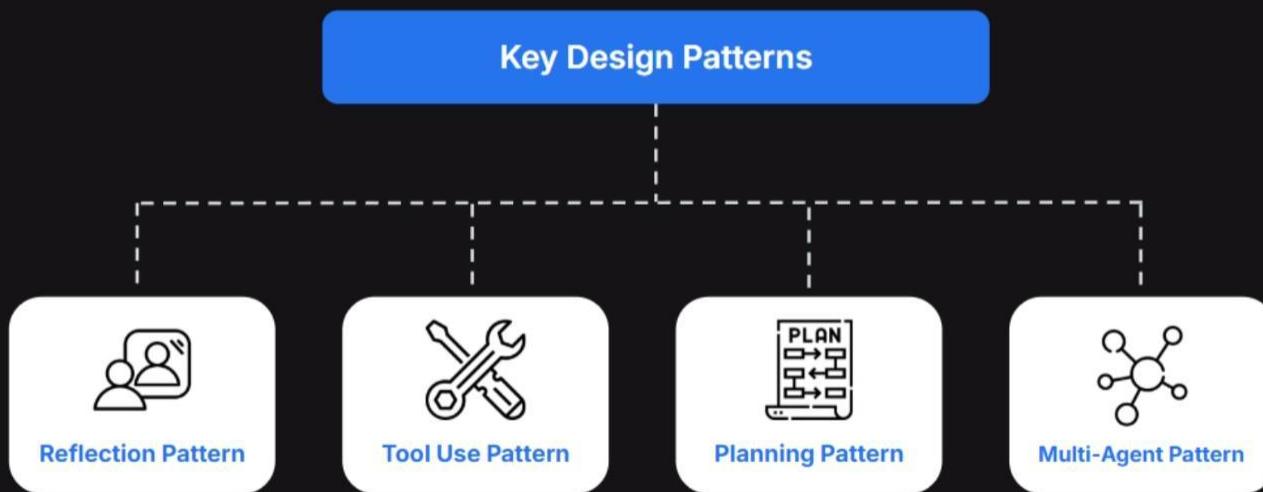
```
ExtractedMemory(
    id="6f555d97-387e-4af6-a23f-a66b4e809b0e",
    content=UserProfile(
        name="Alex",
        preferred_name="Lex",
        response_style_preference="casual and witty with appropriate emojis",
        special_skills=[
            "Python programming",
            "AI development",
            "competitive speedcubing",
        ],
        other_preferences=[
            "prefers informal communication",
            "dislikes corporate-style interactions",
        ],
    ),
)
```

# Key Design Patterns for Agentic AI

Almost a year ago, Andrew Ng defined four design patterns recognizable in Agentic AI Systems



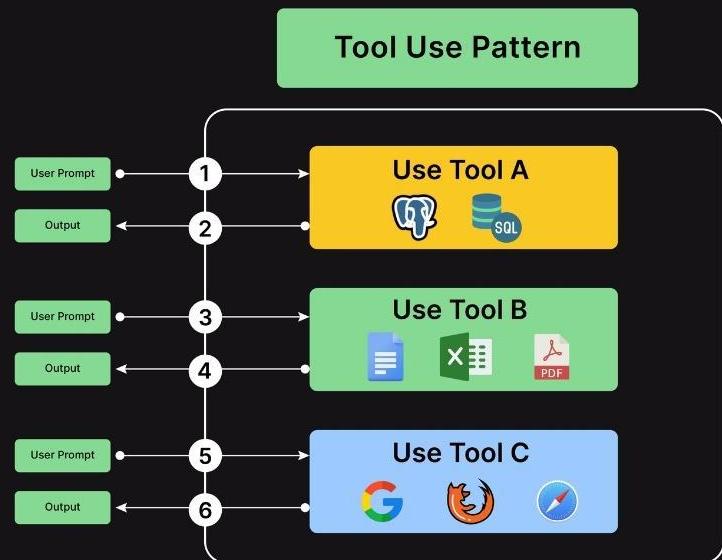
# Key Design Patterns for Agentic AI



# Recommendations for using the Tool Use Pattern

Enables AI Agents to interact with external tools, APIs, and resources for improved functionality and context to support their reasoning.

- These systems can easily handle ~10 tools.
- Can also handle multi-step and multi-tool call executions (ReAct is built-in)
- Best Practices:
  - Well-defined tool schemas for accurate function calling
  - Well-structured System Prompt with detailed instructions
  - Powerful LLMs already trained for function (tool) calling

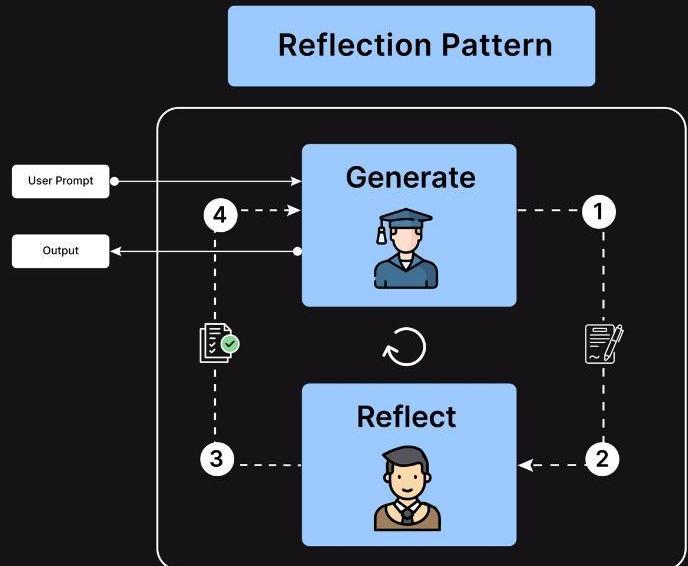


Source: <https://theneuralmaze.substack.com/>

# Recommendations for using the Reflection Pattern

Enables AI Agents to alternate between generating and critiquing for iterative improvement of the generated response.

- Define **clear evaluation criteria** and use only when **iterative refinement** provides **measurable value**.
- Examples:
  - Judging and grading the quality of an LLM response
  - Validating specific guidelines e.g, claims processing
- Best Practices:
  - Use a powerful LLM as a Judge (avoid SLMs)
  - Create well-defined prompts for judging (prefer categories to ranges)
  - Have a max iterations cutoff to prevent infinite loops, besides clear stopping criteria

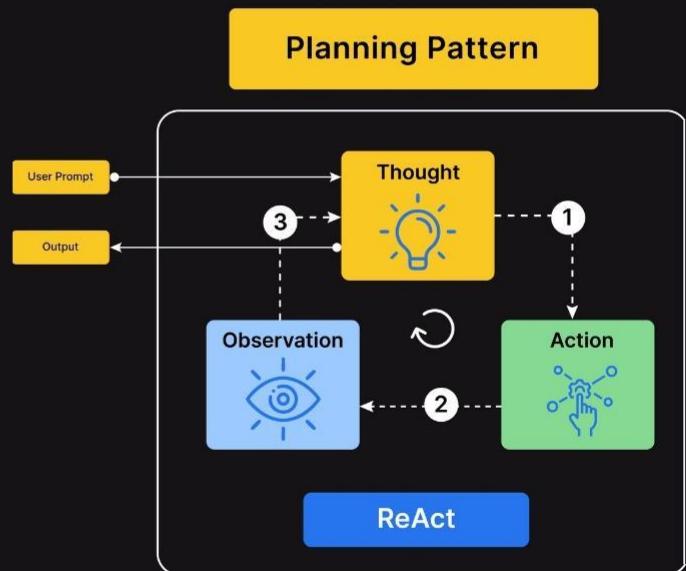


Source: <https://theneuralmaze.substack.com/>

# Recommendations for using the Planning Pattern

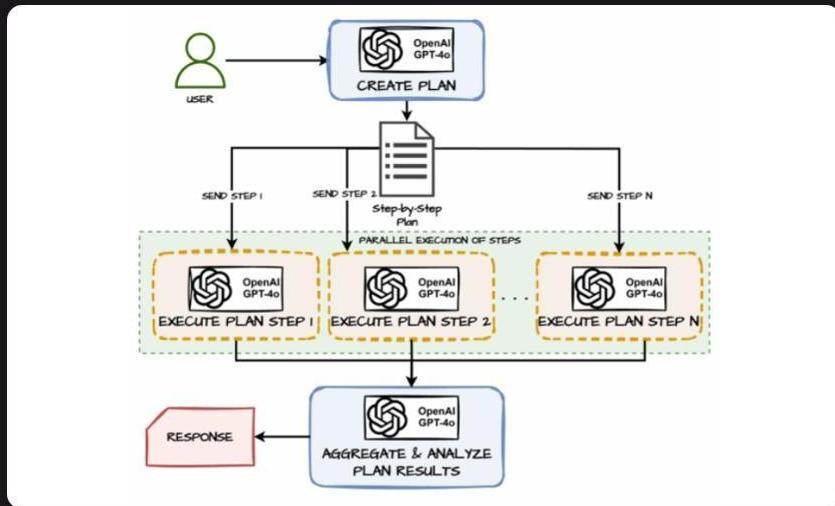
Structures and executes multi-step tasks through reasoning & planning.

- Most ReAct Agents already have planning built-in so first start with simple ReAct Agents
- Best Practices:
  - For more complex tasks, consider adding additional custom planning modules
  - Planning modules or patterns are typically:
    - Static Planners with Parallel Task Execution & Synthesis
    - Dynamic Planners with Task Execution, Reflection & Replanning



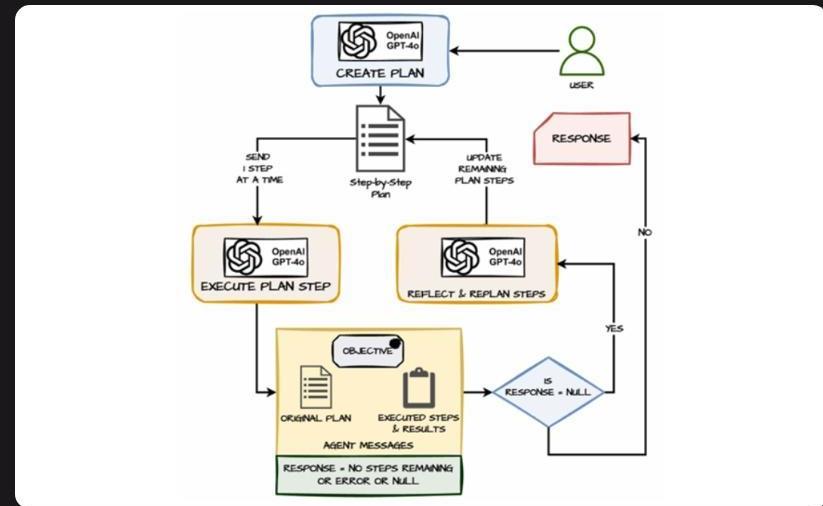
Source: <https://theneuralmaze.substack.com>

# Custom Planning Patterns



## Static Planners

- Break down a task into multiple steps
- Execute all steps in parallel
- Synthesize results from all steps and generate final response (map-reduce)
- Useful when steps do not have dependencies



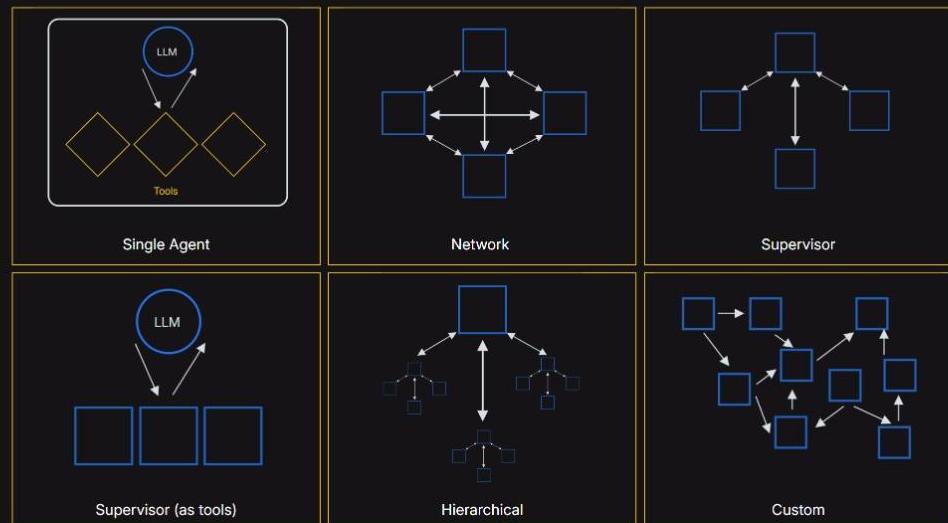
## Dynamic Planners

- Break down a task into multiple steps
- Executes one step at a time
- Reflect and replan remaining steps if needed
- Synthesize results from all steps and generate final response
- Useful when steps have dependencies

# Recommendations for using the Multi-Agent Pattern

Enables multiple AI Agents to solve complex problems through communication and coordination.

- Common architecture patterns:
  - **Network:** Each agent can communicate with every other agent.
  - **Supervisor:** Each sub-agent communicates with a single supervisor agent, which makes decisions.
  - **Hierarchical:** Multi-agent system with a supervisor of supervisors.
- Best Practices:
  - Always start with a simple supervisor or network architecture, and then expand
  - Create separate agents based on specific processes, tasks, tools, and flows



# What is Deep Research?

February 2, 2025 Release

# Introducing deep research

An agent that uses reasoning to synthesize large amounts of online information and complete multi-step research tasks for you. Available to Pro users today, Plus and Team next.

Try on ChatGPT ↗

OpenAI launched [Deep Research](#) on February 2, 2025, where it has been launched as an additional capability in their ChatGPT product. They call this a new agentic capability that can do multi-step research on the internet for complex tasks or queries given by the user. What they claim is that it accomplishes in tens of minutes what would take a human many hours.

# What is Deep Research?

What are you researching?

I need a detailed report on the current strategy Nvidia is following to win the majority markets in AI, do your own research and generate the report without asking me any followup questions



Search

Deep research

...



Got it. I'll generate a detailed report on Nvidia's current strategy to dominate the AI market, covering key elements such as their product offerings, ecosystem, partnerships, and competitive moves. I'll get back to you shortly with a comprehensive breakdown.

Starting research

# What is Deep Research?

Research completed in 8m · 21 sources

## Nvidia's AI Market Domination Strategy (2025)

### Introduction

Nvidia has emerged as the **undisputed leader** in **artificial intelligence (AI) hardware**, powering a majority of AI workloads worldwide. Various estimates indicate Nvidia holds between 80% to 95% of the AI accelerator market [patentpc.com](#) [atobase.com](#), thanks to a holistic strategy spanning cutting-edge **hardware**, a robust **software ecosystem**, deep industry partnerships, strategic acquisitions/investments, and expansion across **global markets**. This report provides a detailed breakdown of Nvidia's current strategy to maintain dominance in AI, covering its GPU and custom silicon roadmap, software platforms (CUDA and beyond), key collaborations (with OpenAI, cloud providers, etc.), acquisitions bolstering AI capabilities, sector-specific initiatives (cloud data centers, enterprise, automotive, healthcare, robotics), competitive positioning against rivals, and global expansion.

### Hardware Strategy: GPUs, Data Center Products, and Custom Silicon

Nvidia's hardware strategy centers on relentless GPU innovation and specialized processors for AI:

- Flagship GPUs for AI: Nvidia's latest data center GPUs like the **A100** (2020, Ampere architecture) and **H100** (2022, Hopper architecture) have become industry standards for training and deploying AI models. The H100 in particular introduced a Transformer Engine and FP8 precision, offering up to **9x faster training** and **30x faster inference** on large language models compared to its predecessor A100 [nvidianews.nvidia.com](#). These GPUs are highly sought after – an H100 unit can cost \$25,000–\$40,000 due to surging demand [patentpc.com](#). Nvidia is already pushing forward with next-generation GPUs (Blackwell architecture slated for 2024) to sustain performance leadership, with the upcoming "**GB200**" chip reportedly delivering up to **30x** H100's performance on LLM inference at far lower energy/cost [investopedia.com](#).
- Grace Hopper "Superchip" (GH200): To handle ever-larger AI models, Nvidia introduced the **Grace Hopper** (GH200) Superchip, which tightly couples a 72-core Arm-based Grace CPU with an H100 GPU via high-speed NVLink. This CPU-GPU fusion provides massive memory capacity and bandwidth – in dual-chip configuration, GH200 offers 144 CPU cores, 1.2 TB of unified memory, and 10 TB/s bandwidth, allowing generative AI models **3.5x** larger to be trained/inferenced than previous systems [nvidianews.nvidia.com](#) [nvidianews.nvidia.com](#). The GH200 is optimized for giant-scale AI and HPC workloads, effectively a custom silicon platform for data centers in the AI era.
- Integrated AI Systems (DGX & HGX): Nvidia packages its chips into full systems like the **DGX series** – ready-to-use AI supercomputers. For example, the DGX H100 integrates 8x H100 GPUs with high-speed NVLink and InfiniBand interconnects, delivering 32 petaFLOPs of AI compute in one node [nvidianews.nvidia.com](#). Such systems come with Nvidia's complete software stack, enabling enterprises to plug-and-play AI at scale [nvidianews.nvidia.com](#). Nvidia also provides the **HGX server boards** to OEMs, standardizing AI server designs. This vertical integration (from silicon to systems) ensures optimal performance and quicker adoption in data centers.

### Software Ecosystem: CUDA, Frameworks, and Developer Platforms

A critical pillar of Nvidia's AI dominance is its expansive software ecosystem. CUDA, Nvidia's parallel computing platform introduced in 2006, is often cited as the *main reason* competitors struggle to catch up [patentpc.com](#). CUDA provides a common language (APIs and libraries) that lets developers harness GPU acceleration for general-purpose computing. Over years, Nvidia built a rich stack of AI software on top of CUDA:

- CUDA and Developer Libraries: The CUDA toolkit and drivers enable GPUs to accelerate popular programming languages (C++, Python via CUDA Python, etc.). Nvidia continuously updates **cuDNN** (CUDA Deep Neural Network library) for training neural nets, **TensorRT** for high-speed inference optimization, and countless other libraries (cuBLAS, NCCL for multi-GPU communication, etc.). This ensures that AI models run efficiently on Nvidia hardware out-of-the-box. Developers have collectively invested millions of hours in CUDA code – forming a moat that [makes switching to alternative GPU platforms costly](#).
- AI Framework Integrations: Nvidia works closely with major frameworks like PyTorch, TensorFlow, and JAX to optimize them for Nvidia GPUs. Many frameworks (TorchVision or TensorFlow ops) use Nvidia's CUDA libraries **containers** (NGC) to provide ready-to-run environments for deployment. The result is that data scientists can train models on Nvidia hardware without writing custom code or less between hardware and software has made Nvidia's platform [development](#) [patentpc.com](#).

### Partnerships and Collaborations Driving Dominance

Nvidia amplifies its reach through **strategic partnerships** with leading AI players and tech companies. These collaborations ensure Nvidia's technology is deeply embedded in the AI infrastructure of cloud providers, research labs, and enterprises:

- Cloud Hyperscalers (Microsoft, AWS, Google, Oracle): All major cloud providers rely on Nvidia GPUs to power their AI services:
  - Microsoft (Azure): Nvidia and Microsoft announced a multi-year collaboration to build a massive AI supercomputer in Azure, combining tens of thousands of Nvidia GPUs with Nvidia's high-speed networking and software stack [techcrunch.com](#). Azure's latest GPU instances (ND H100 v5) are based on Nvidia H100 NVL platforms [azure.microsoft.com](#). Notably, **OpenAI's models** (like ChatGPT and GPT-4) are trained on Azure, using **Nvidia A100/H100 GPUs** in large clusters [nvidianews.nvidia.com](#). Microsoft's partnership with Nvidia ensures Azure remains a top choice for AI workloads, offering state-of-the-art Nvidia hardware to its customers.
  - Amazon Web Services (AWS): AWS has long offered Nvidia GPU instances (P3, P4, P5 instances, etc.) for AI. In 2023 AWS announced EC2 UltraClusters with up to **20,000 H100 GPUs interconnected** for extreme-scale training [nvidianews.nvidia.com](#). Nvidia works with AWS to integrate features like **Nvidia's Quantum-2 InfiniBand** in these clusters, enabling near-supercomputer performance in the cloud. While AWS has developed its own AI chips (Trainium for training, Inferentia for inference) to reduce cost, the AWS CEO characterized their in-house chips as a "supplement" to Nvidia GPUs, not a full replacement, given Nvidia's broad capabilities [reidelt.com](#). In practice, Nvidia GPUs still power many of the highest-end AI jobs on AWS, and AWS's support of the Nvidia ecosystem (e.g. selling Nvidia's DGX systems via AWS Marketplace) shows a cooperative stance.
  - Google Cloud: Google famously designed TPUs (Tensor Processing Units) for its internal AI needs, yet Nvidia remains an important partner. Google Cloud offers Nvidia A100 and H100 instances to customers, and even Google's own AI research uses Nvidia hardware for certain projects. For instance, Google's next-gen **Gemini** AI model is reportedly being trained on Nvidia GPUs, despite

# Build your own Deep Research Agent



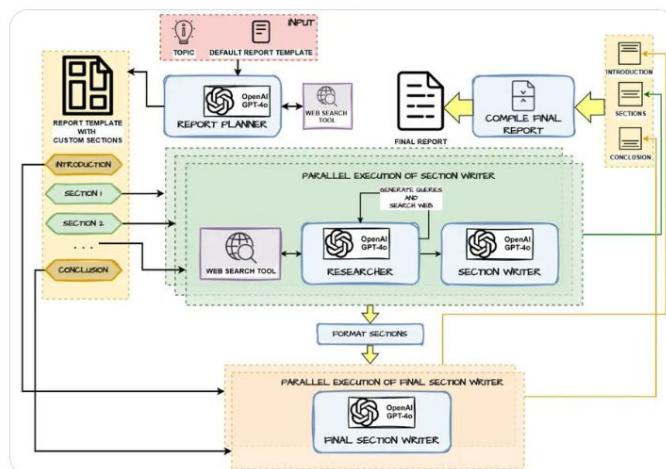
...



An autonomous research agent built with LangGraph that combines GPT-4 and Tavily AI to perform parallel research and generate structured reports, all through sophisticated state management.

🚀 Check out the implementation:

[analyticsvidhya.com/blog/2025/02/b...](https://analyticsvidhya.com/blog/2025/02/b...)



Check out the [detailed blog here](#)

1:30 AM · Mar 10, 2025 · 45.2K Views

6

181

726

613

↑

# Build your own Deep Research Agent

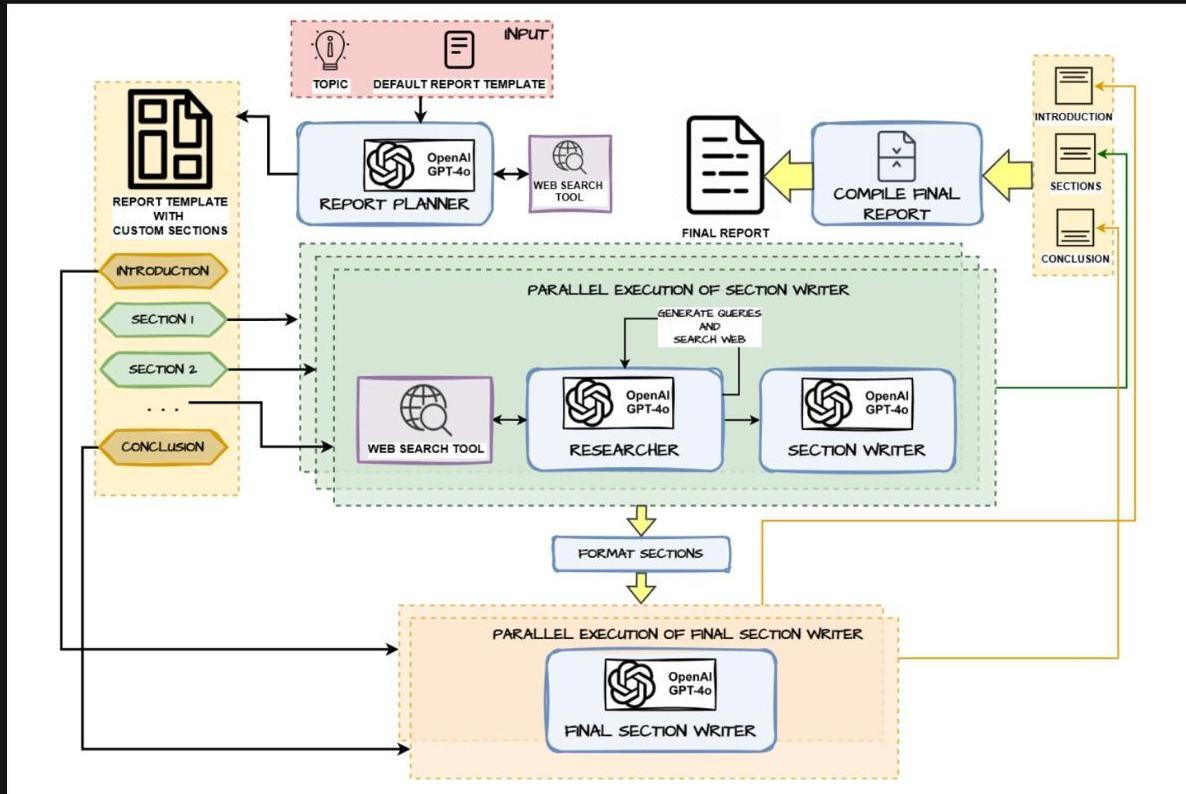
## Goal

Build an AI agent that performs deep topic research and generates **structured, wiki-style reports** using GPT-4o and LangGraph.

## Key Features

- Custom report planning using GPT-4o
- Parallel web research per section
- Structured section writing
- Intro & Conclusion synthesis
- Final markdown report generation

# Planning Agent for Deep Research & Structured Report Generation Architecture



# Build your own Deep Research Agent

## How It Works

1.  User inputs topic + default template
2.  GPT-4o plans report sections
3.  Parallel execution for each section:
  -  Web search (Tavily)
  -  Section content generation
4.  Formatting & markdown cleanup
5.  Final intro/conclusion generation
6.  Full report compilation

# Build your own Deep Research Agent

Each section undergoes this process:

-  **Researcher** → Generates queries & collects results
  -  **Web Search Tool** → Executes queries
  -  **Section Writer** → Writes clean, structured content
-  All steps happen **in parallel** using LangGraph's distributed node execution!

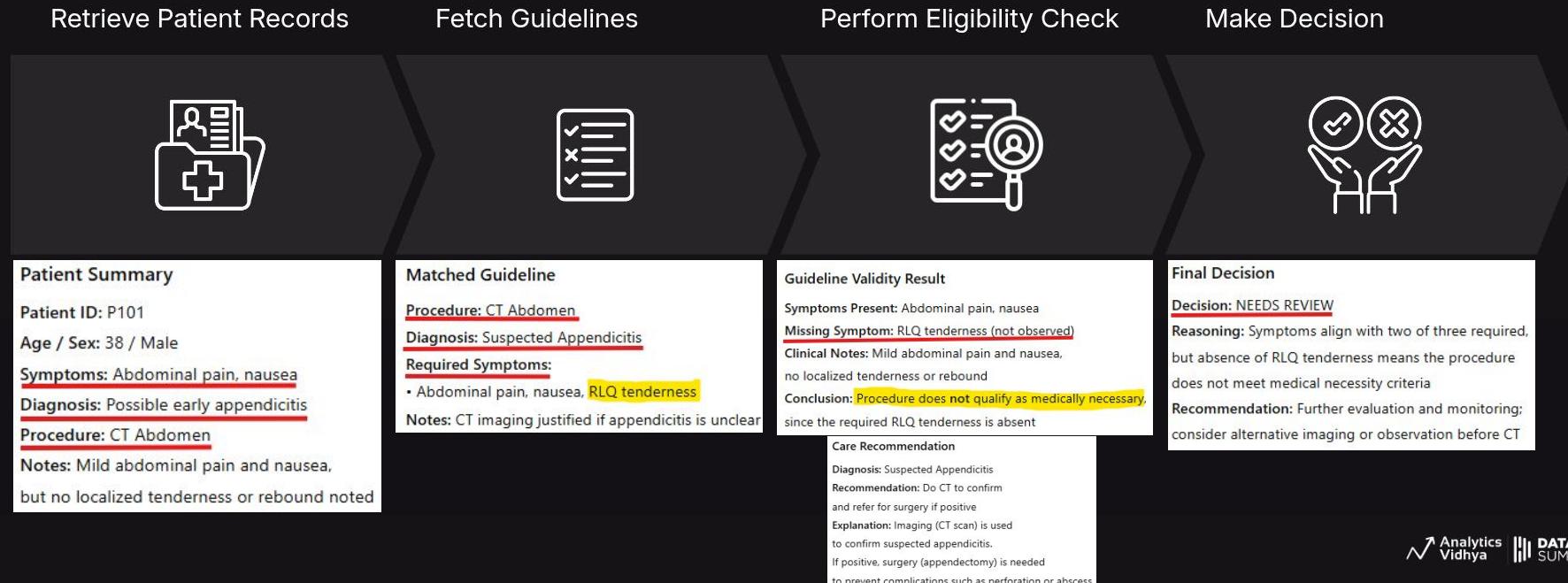
## Formatting + Final Sections

-  Applies markdown cleanup and escape sequences
-  GPT-4o creates the **Introduction** and **Conclusion** based on core sections
-  All parts are stitched into a clean, readable report

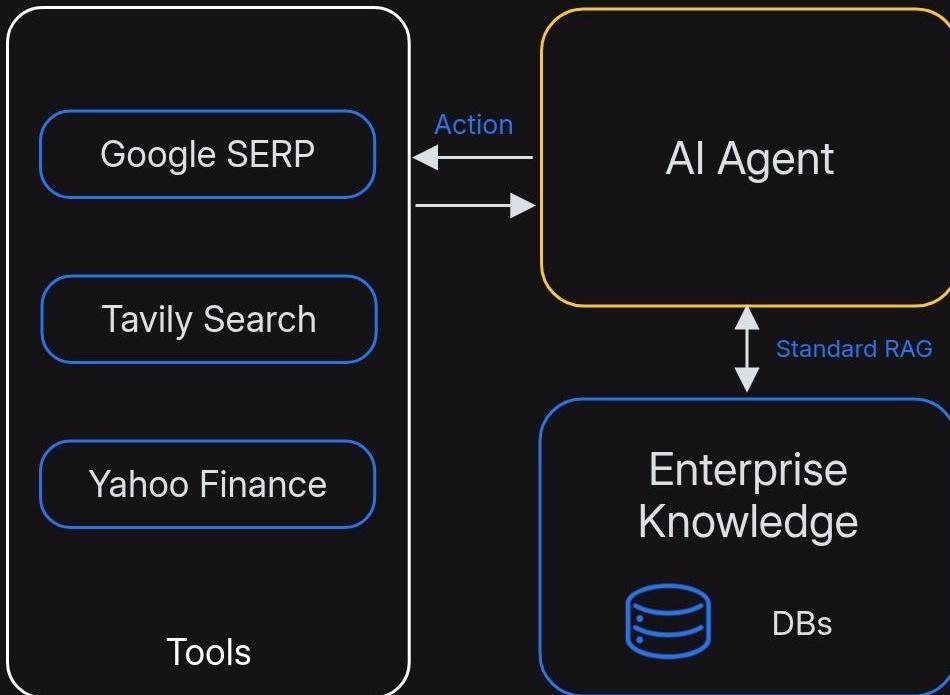
# Let's look at a Real-World Use-Case

## Utilization Review

Utilization review is the process of evaluating patient medical procedures to ensure they are necessary, appropriate, and aligned with clinical guidelines and insurance coverage policies.



# RAG + AI Agents = Agentic RAG

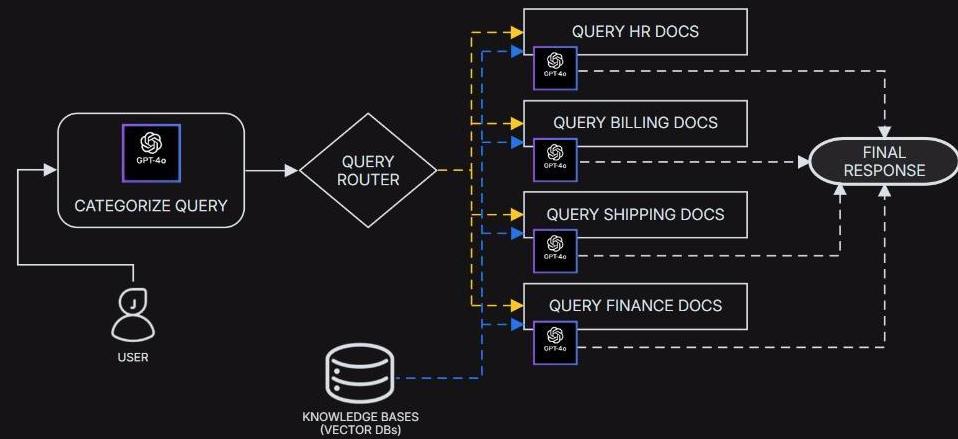


- Agentic RAG is a combination of AI Agents and RAG Systems
- AI Agents helps in reasoning, planning, calling tools
- RAG helps the Agent in getting contextual data from enterprise knowledge bases
- Examples include Router RAG, Self-Reflective RAG and more

# Router Agentic RAG Systems = Routing Agents + RAG

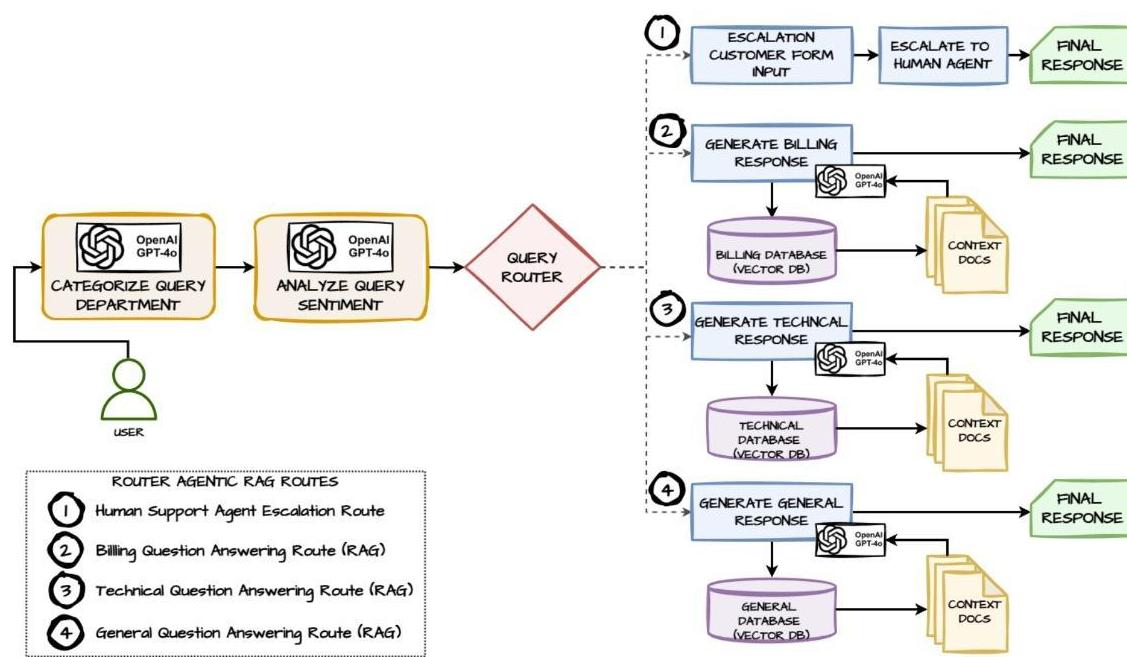
The main components of this system include:

- LLM-routing mechanisms to route user queries to specific workflows or sub-agents
- Each sub-agent or workflow handles queries of a specific type using RAG

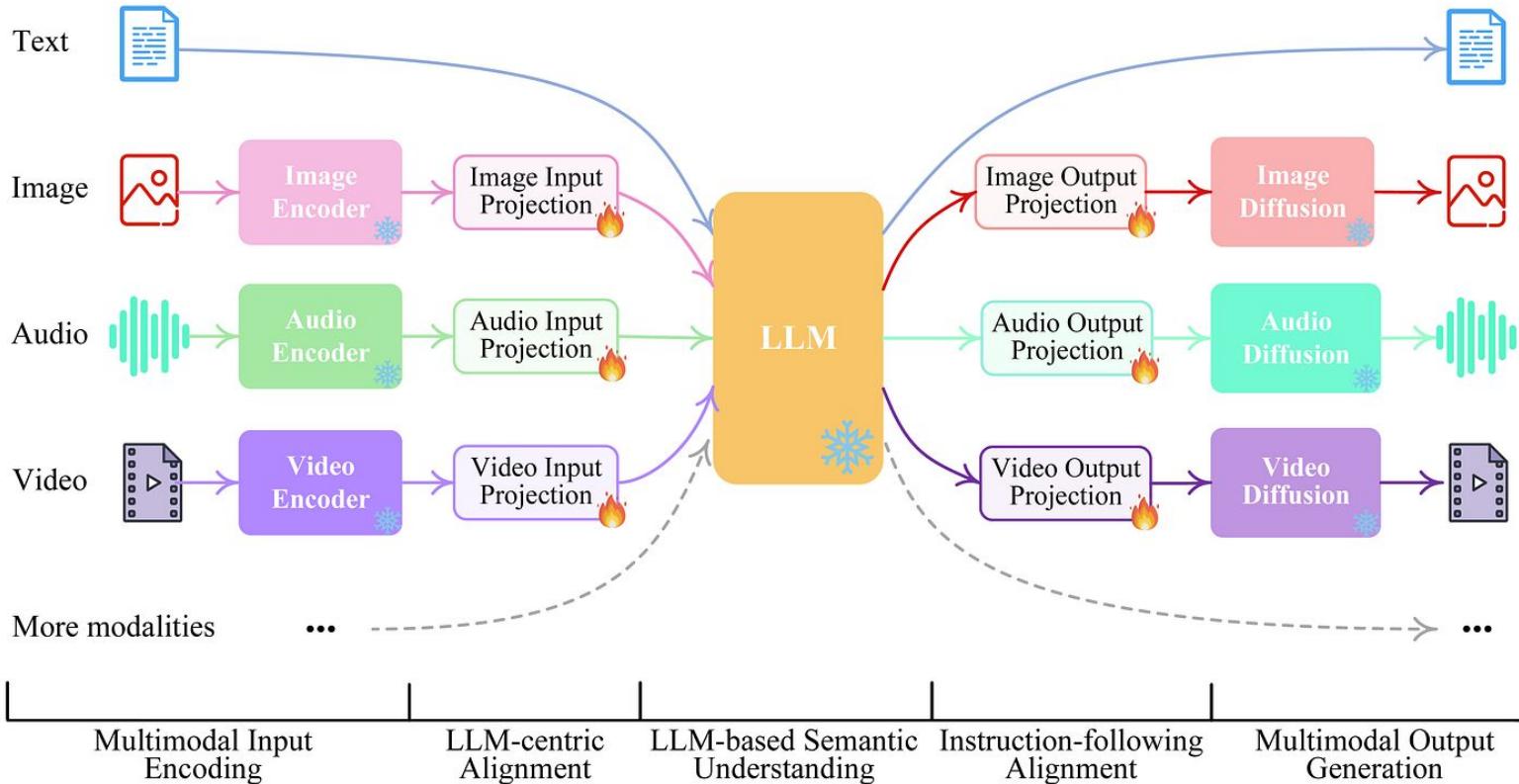


# Let's Look at a Real-World Use-Case

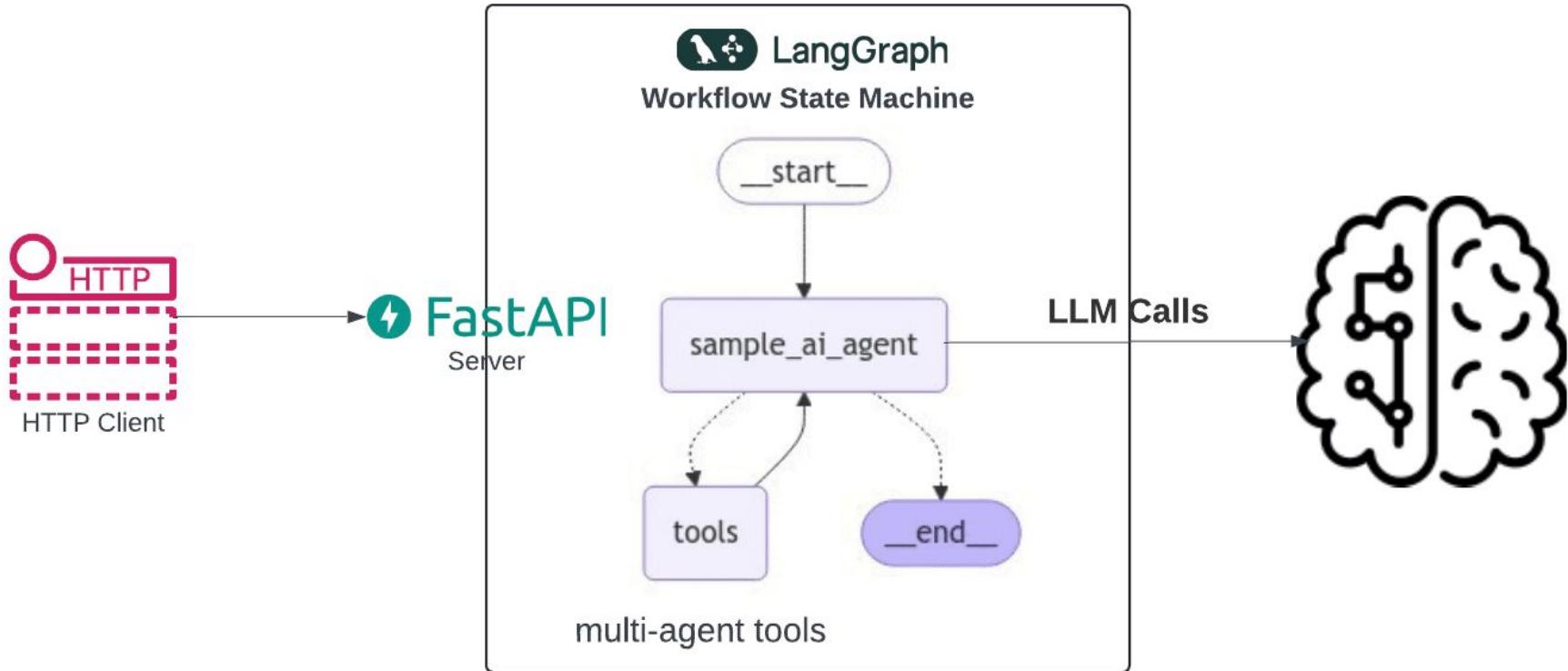
## Customer Support Automation



# Multi-Modal LLMs

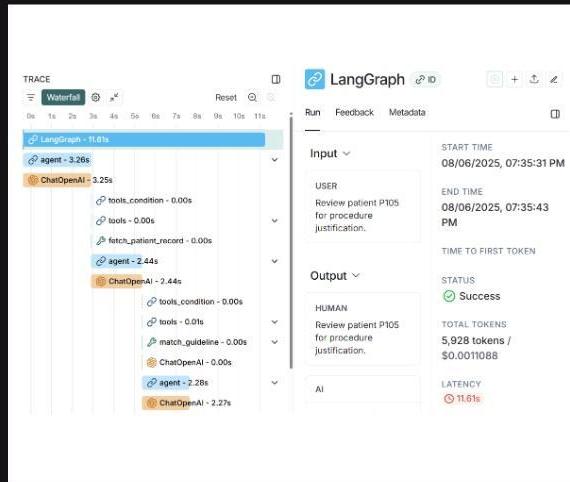


# Wrap your Agent in an API and Deploy

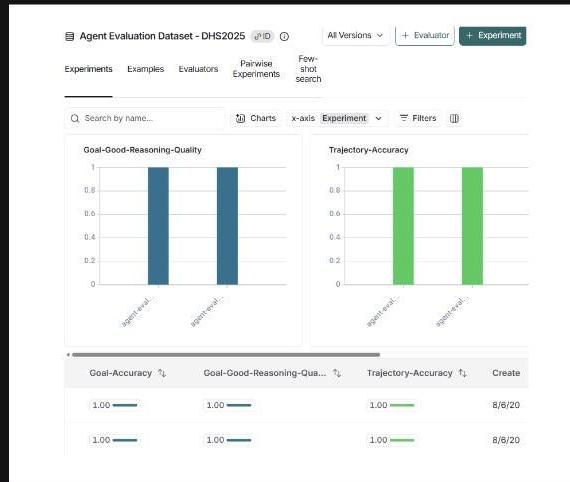


# Why Care About Agent Observability?

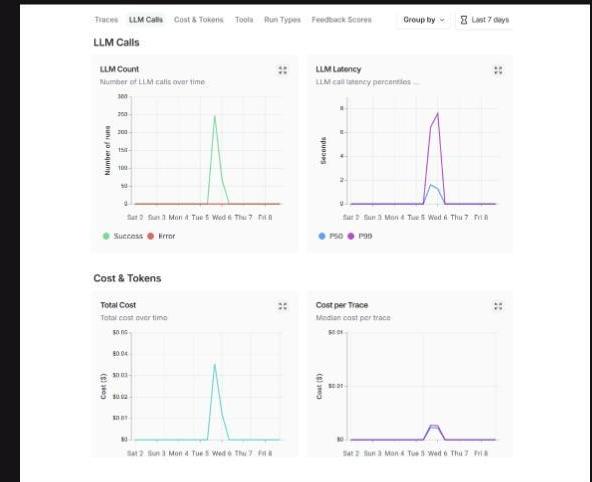
Observability enables you to take your Agent from PoC to Production and beyond with tracing, debugging, evaluation and continuous monitoring.



**Trace Agent Executions** for debugging, explainability, audits and more



**Evaluate your Agent's Performance** with LLM-as-a-Judge & Statistical Evaluators

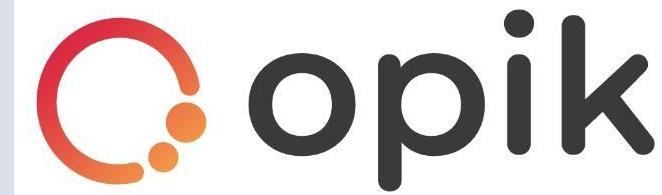


**Monitor and Track your Agent's Usage** with cost, latency, usage metrics and more

# Popular Frameworks for Observability in Agentic AI Systems



LangSmith



by Comet



# Langfuse

confident-ai/  
deepeval

The LLM Evaluation Framework



184  
Contributors

913  
Used by

32  
Discussions

10k  
Stars

872  
Forks



# Common Metrics for Agent Monitoring

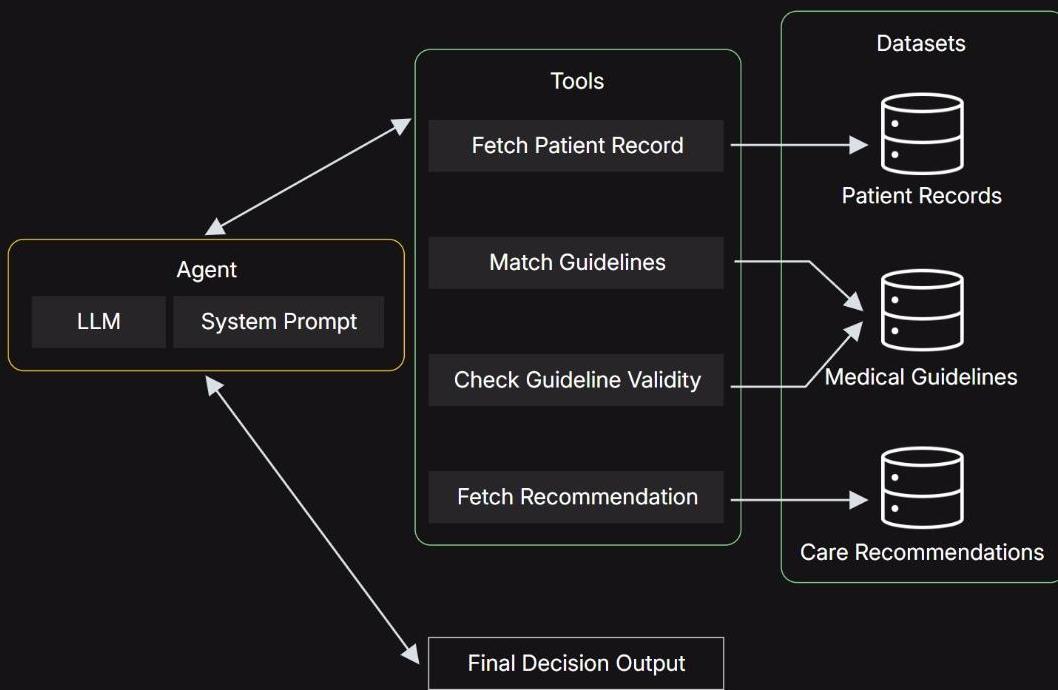
Category	Main Metric	Meaning
Traces	Trace count	Number of complete agent operation traces recorded.
	Latency (avg, p50, p99)	Time to complete a trace; percentiles show median and tail performance.
	Error rate	Percentage of traces that failed.
LLM Calls	Call count	Number of LLM invocations in traces.
	Call latency (avg, p50, p99)	Time taken per LLM call; percentiles indicate slowest cases.
Cost & Tokens	Cost	Monetary cost from token usage.
	Token breakdown	Split of prompt vs. completion tokens.
Tools	Tool run count	Frequency of tool executions.
	Tool latency (avg, p50, p99)	Time taken by tools; percentiles highlight tail latencies.
	Tool error rate	Percentage of tool runs that failed.

# Common Metrics for Agent Evaluation

Recommendation: Create your own custom LLM-as-a-Judge metric prompts and avoid relying on default prompts from frameworks. Use few-shot examples and also develop an evaluation dataset; it really helps!

Metric	Description
Goal Accuracy	Measures whether the agent achieves the intended goal, using reference comparison or LLM-as-a-judge when no reference exists.
Goal Reasoning Quality	Evaluates how well the agent's reasoning supports the final decision, focusing on clarity, correctness, and alignment with the goal.
Trajectory Accuracy	Checks if the agent follows the correct sequence of steps - approaches include LLM-as-a-Judge, Strict Match, Unordered Match, and Subset/Superset Match.
Tool Call Accuracy	Assesses whether the correct tools are selected and used with appropriate inputs during execution.
Hallucination	Measures if the agent outputs unsupported or fabricated information.

*Let's now add observability to our Utilization Review Agent from earlier!*



## Build Agent

**TRACE**

Waterfall | Reset | 0s 1s 2s 3s 4s 5s 6s 7s 8s 9s 10s 11s

LangGraph - 11.0s

agent - 3.26s

- tools\_condition - 0.00s
- tools - 0.00s
- fetch\_patient\_record - 0.00s
- agent - 2.44s

ChatOpenAI - 3.25s

- tools\_condition - 0.00s
- tools - 0.00s
- match\_guideline - 0.00s
- ChatOpenAI - 2.44s

HUMAN

Review patient P105 for procedure justification.

**Monitor Agent**

Run Feedback Metadata

START TIME: 08/06/2025, 07:35:31 PM

END TIME: 08/06/2025, 07:35:43 PM

TIME TO FIRST TOKEN: 08:06:00

STATUS: Success

TOTAL TOKENS: 5,928 tokens / \$0.0011088

LATENCY: 11.61s

## Monitor Agent

