# Let us learn Agentic AI and build AI Copilots in 3 hours!

# Workshop Agenda:

## 1. Introduction (30 minutes)

**What is Agentic AI?**:

- Definition and key characteristics.
- How it differs from traditional AI.
- Examples of Agentic AI in action (e.g., autonomous agents, AI Copilots).

**Why AI Copilots?**: (Uday)

- Role of AI Copilots in enhancing productivity and decision-making.
- Real-world examples (e.g., GitHub Copilot, Microsoft 365 Copilot).

## 2. Core Concepts of Agentic AI (30 minutes)

**Key Components of Agentic AI**:

- Autonomy, adaptability, and goal-oriented behaviour.
- Interaction with environments and users.

**Architecture of AI Copilots**:

- Natural Language Processing (NLP) and Large Language Models (LLMs).
- Integration with APIs and external tools.
- Feedback loops and continuous learning.

**Demo**: Show a live example of an AI Copilot in action (e.g., a chatbot or coding assistant). - (Amit, AI Cost Demo)

# What is Agentic AI?

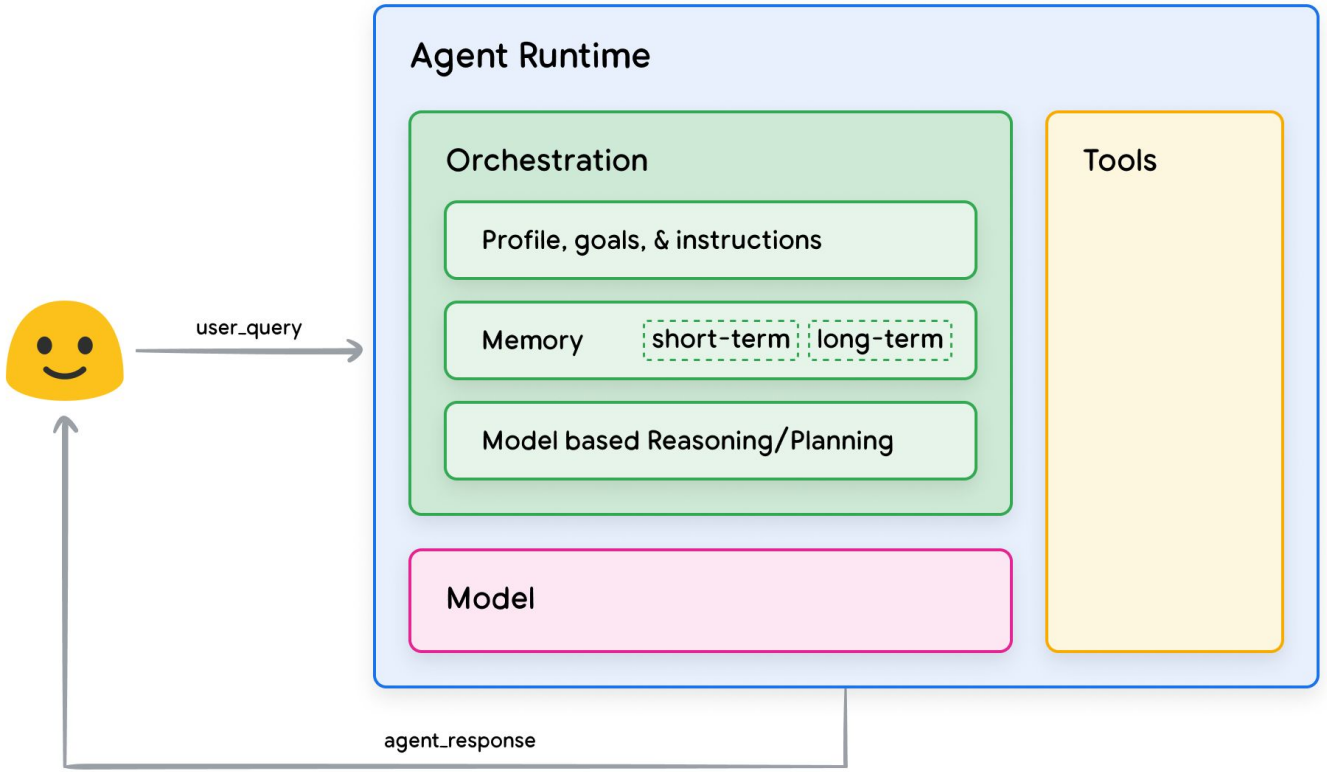
## What is an Agent…

### An Empowered Application Capable of Observation and Action

**Key Characteristics:**

- **Autonomous Operation:** Agentic AI systems can independently analyse data, make decisions, and perform tasks to achieve predefined goals.
- **Adaptive Learning:** These systems continuously learn from interactions and adapt to new situations, enhancing their performance over time.
- **Complex Problem-Solving:** Agentic AI utilizes sophisticated reasoning and iterative planning to autonomously solve complex, multi-step problems.
- **Diverse Applications:** Agentic AI is employed across various sectors, including customer support automation, enterprise workflows, cybersecurity, and business intelligence.
- **Enhanced Productivity:** By automating tasks and making independent decisions, agentic AI systems can significantly boost efficiency and reduce the need for human oversight

# General agent architecture and components



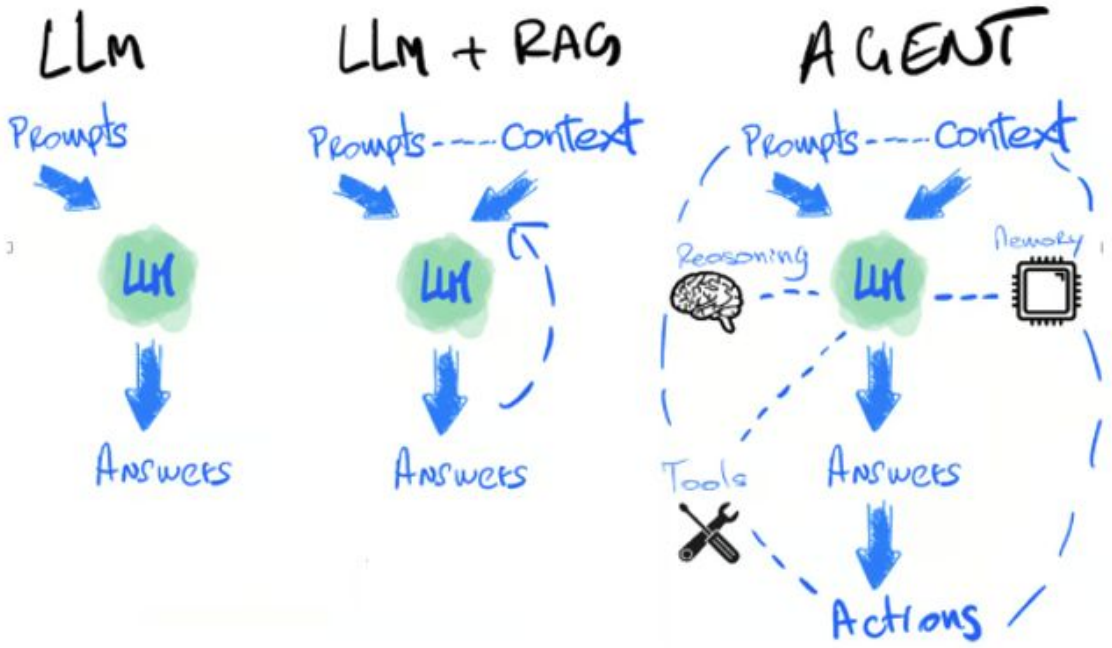

- The Model : In an agent context, the “model” is the language model (or set of models) acting as the centralized decision-maker, capable of following reasoning frameworks like Re-Act or Chain-of-Thought. It can be general purpose or fine-tuned for your application, but is typically not trained with the agent’s specific tool/logic configuration. However, you can refine it further by providing examples that showcase the agent’s unique capabilities and tool usage.

- The Tools : Foundational AI models excel in generating text and images but lack the capability to interact with external data and services. Integrating tools enables these models to perform actions like updating databases or retrieving real-time information, significantly expanding their functionality. This integration supports advanced applications such as retrieval-augmented generation (RAG), enhancing the model's ability to process and utilize real-world data.

- The orchestration layer : The orchestration layer manages an agent's cyclical process of receiving information, reasoning, and making decisions until achieving its goal. Its complexity varies, ranging from simple rule-based calculations to advanced logic chains and machine learning algorithms. This layer ensures seamless coordination among various AI components to accomplish tasks effectively.

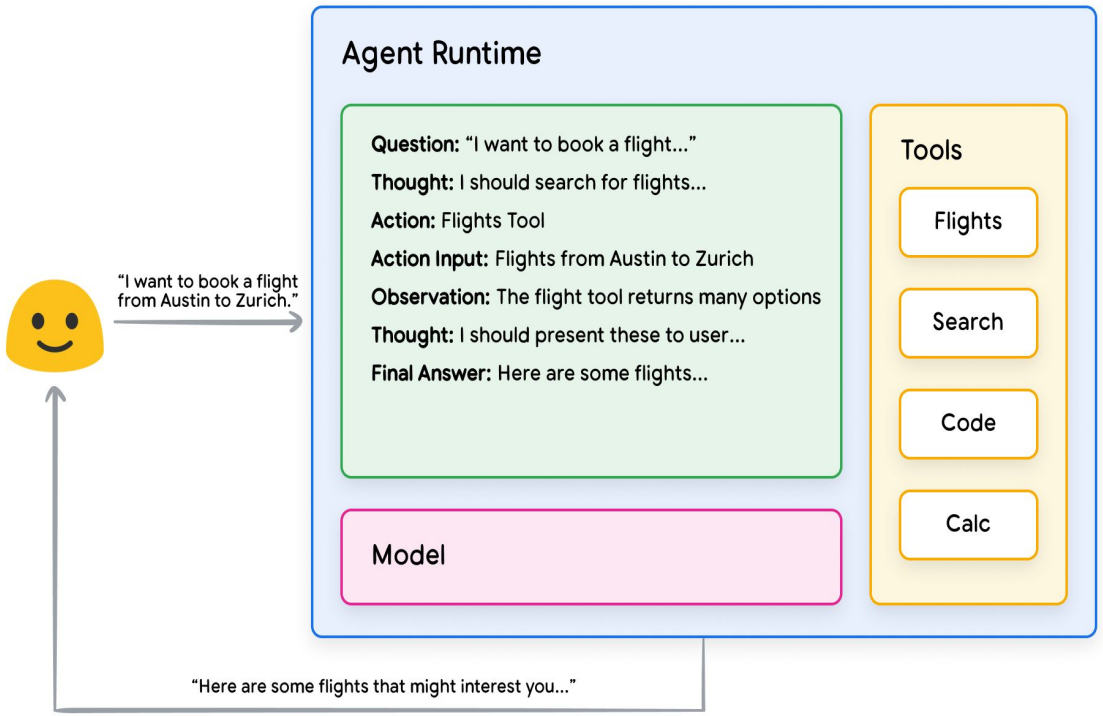
# Agents vs. models

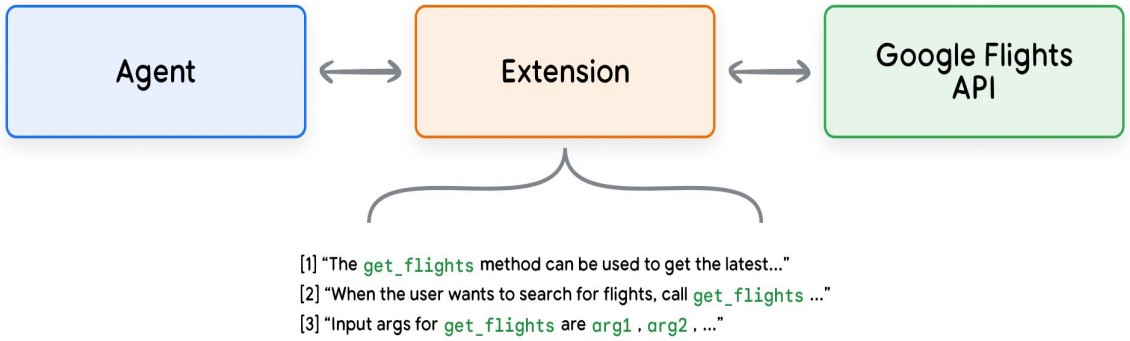
## How Agentic AI systems Differ?




| Models | Agents |
|---|---|
| Knowledge is limited to what is available in their training data. | Knowledge is extended through the connection with external systems via tools |
| Single inference / prediction based on the user query. Unless explicitly implemented for the model, there is no management of session history or continuous context. (i.e. chat history) | Managed session history (i.e. chat history) to allow for multi turn inference / prediction based on user queries and decisions made in the orchestration layer. In this context, a 'turn' is defined as an interaction between the interacting system and the agent. (i.e. 1 incoming event/ query and 1 agent response) |
| No native tool implementation. | Tools are natively implemented in agent architecture. |
| No native logic layer implemented. Users can form prompts as simple questions or use reasoning frameworks (CoT, ReAct, etc.) to form complex prompts to guide the model in prediction. | Native cognitive architecture that uses reasoning frameworks like CoT, ReAct, or other pre-built agent frameworks like LangChain. |

# Examples of Agentic AI in action

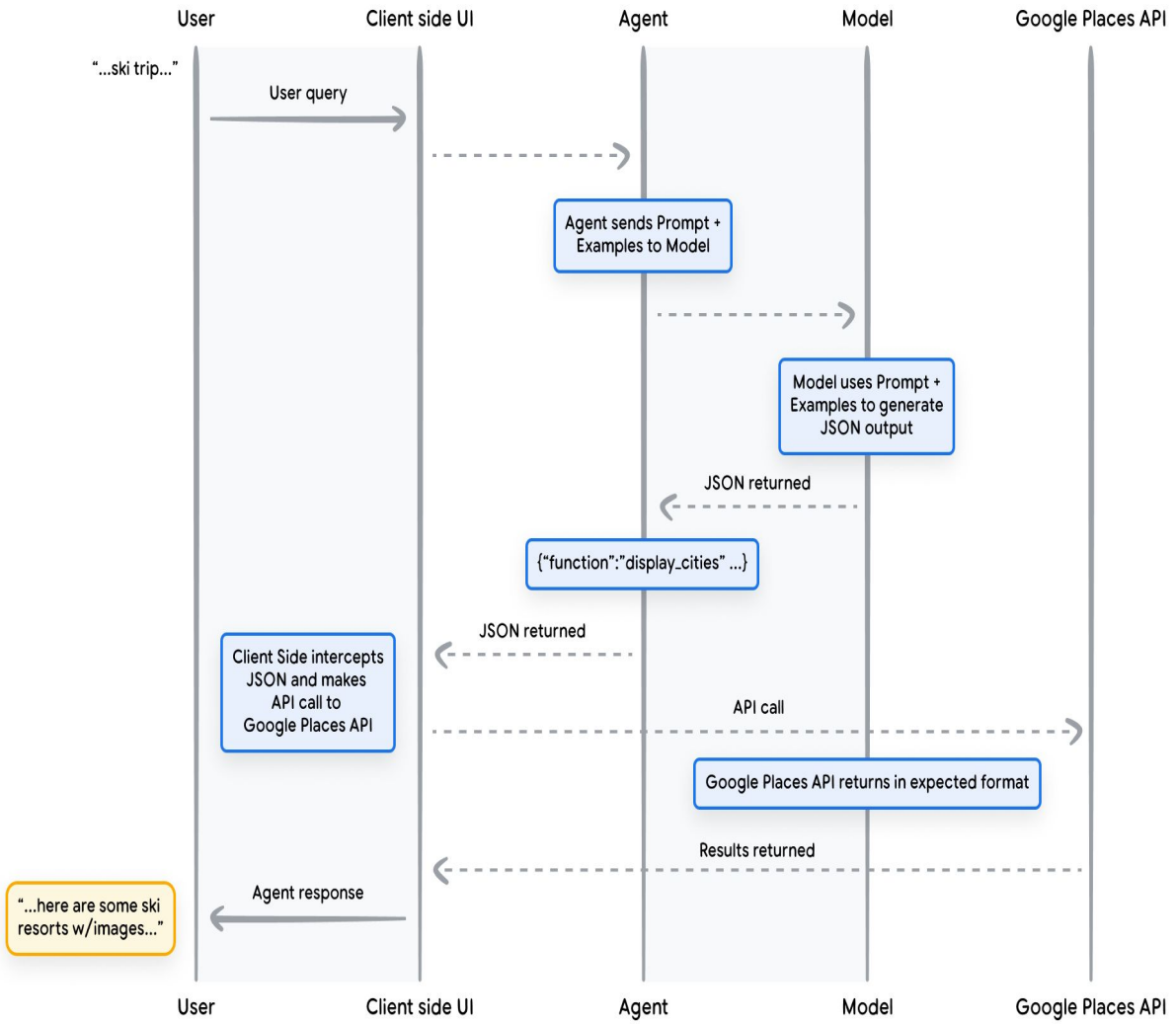


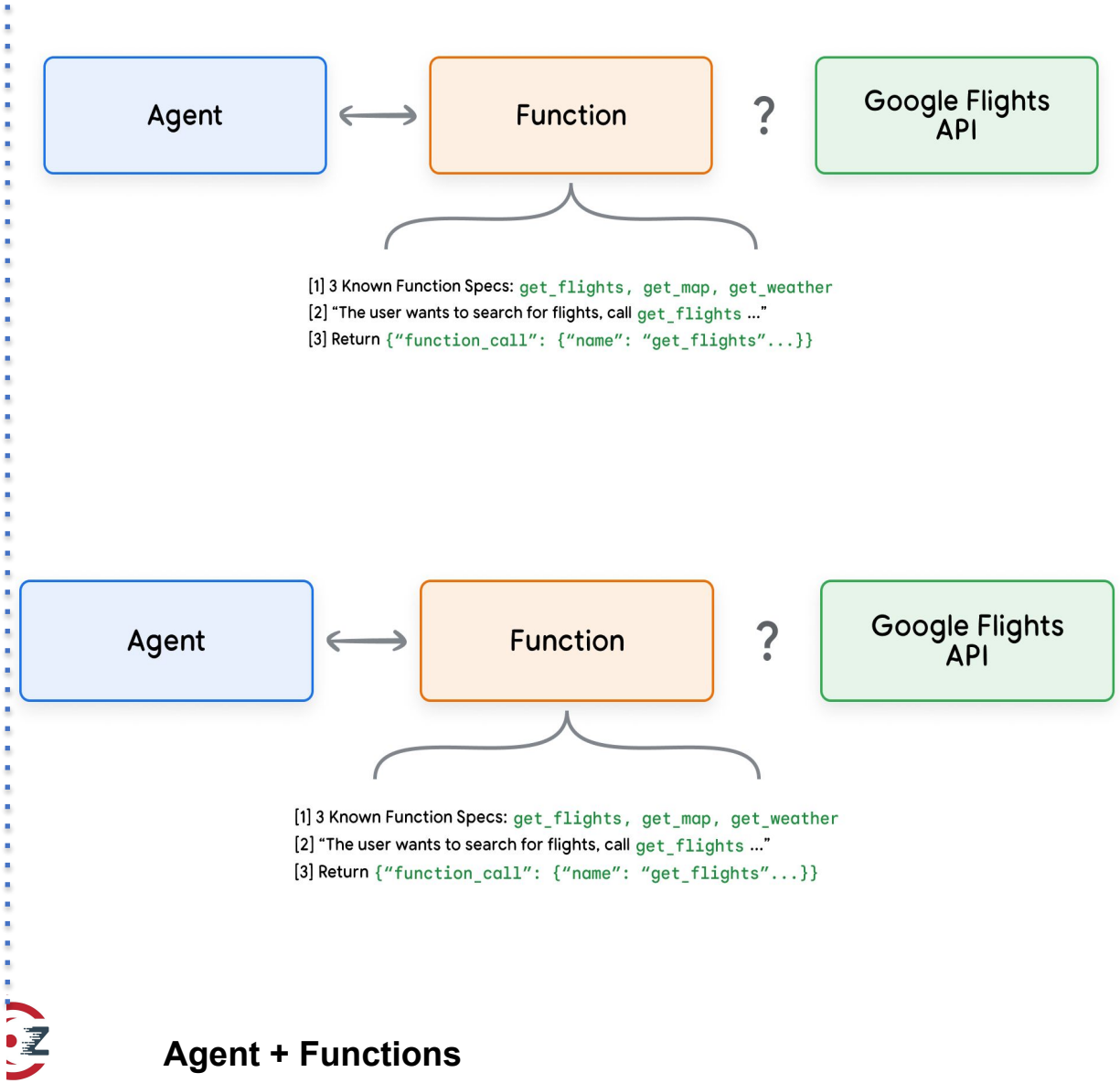





**Agent + Extensions**

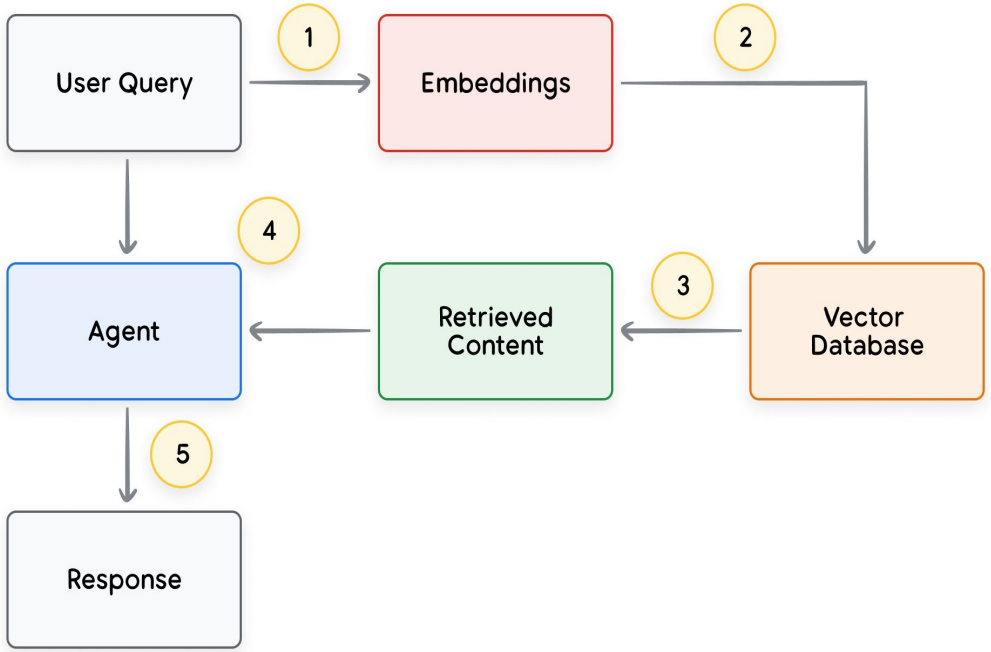
# Examples of Agentic AI in action

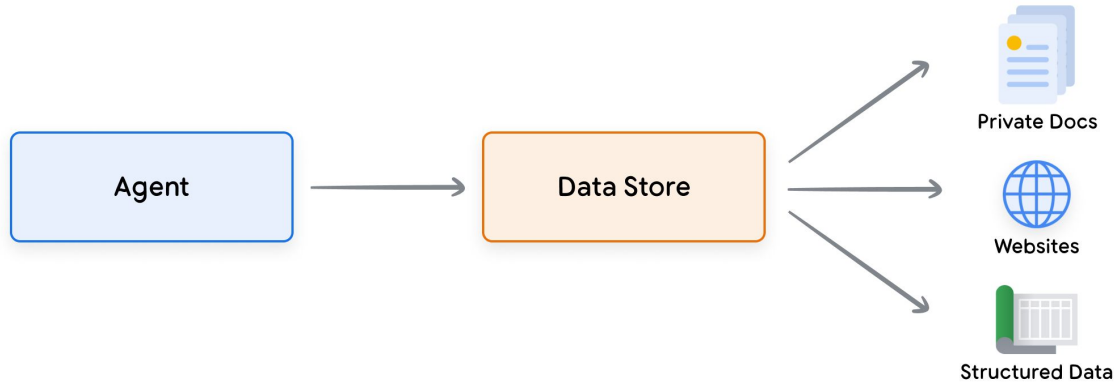



**Agent + Functions**

# Examples of Agentic AI in action



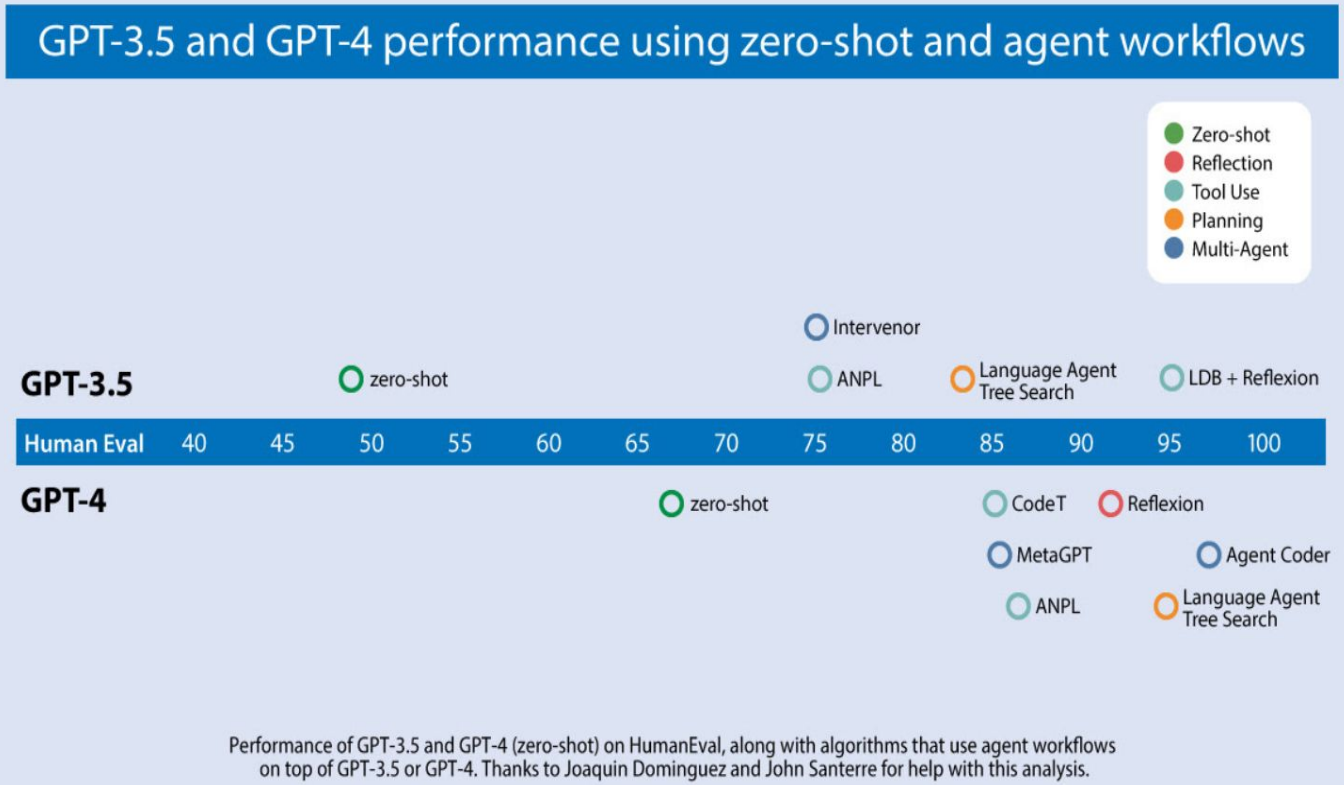





**Agent + Data Store**

# Why Agentic AI?

## Why AI Co-pilits..




Performance of GPT-3.5 and GPT-4 (zero-shot) on HumanEval, along with algorithms that use agent workflows on top of GPT-3.5 or GPT-4. Thanks to Joaquin Dominguez and John Santerre for help with this analysis.

- LLMs in zero-shot mode - prompting a model to generate final output token by token without revising its work. This is akin to asking someone to compose an essay from start to finish, typing straight through with no backspacing allowed, and expecting a high-quality result. Despite the difficulty, LLMs do amazingly well at this task!
- GPT-3.5 (zero shot) was 48.1% correct. GPT-4 (zero shot) does better at 67.0%. However, the improvement from GPT-3.5 to GPT-4 is dwarfed by incorporating an iterative agent workflow. Indeed, wrapped in an agent loop, GPT-3.5 achieves up to 95.1%.
- Agents extend the capabilities of language models by leveraging tools to access real-time information, suggest real-world actions, and plan and execute complex tasks autonomously. agents can leverage one or more language models to decide when and how to transition through states and use external tools to complete any number of complex tasks that would be difficult or impossible for the model to complete on its own.

# Why Agentic AI?

## RPA vs. APA

🤖 **RPA (Robotic Process Automation)**

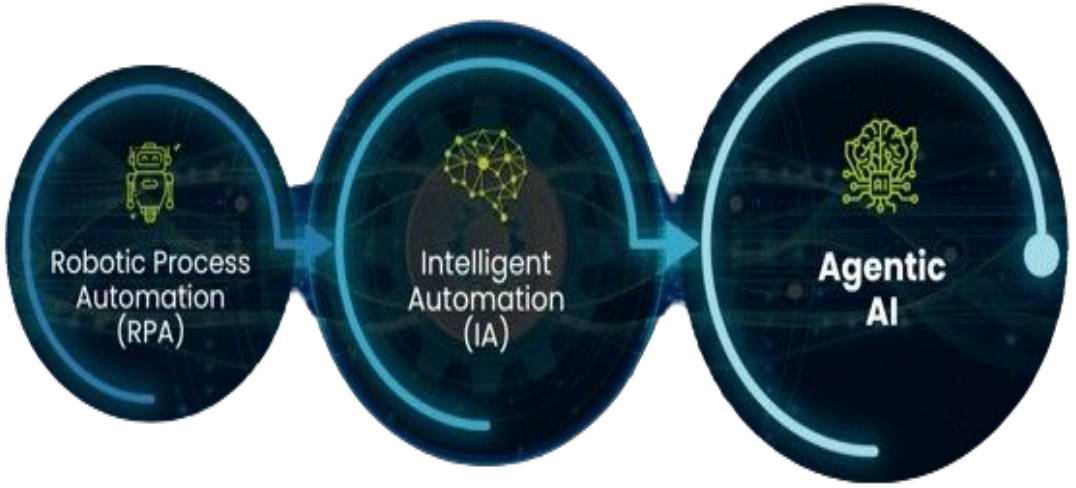
- **How it works:** Executes predefined, rule-based tasks in a fixed sequence.
- **Best for:** Repetitive, manual, and structured tasks such as data entry, form filling, or invoice processing.
- **Limitations:** No learning or adaptation; cannot handle unstructured data or dynamic workflows.
- **Analogy:** Like a macro on steroids – performs tasks exactly as instructed, step by step.

🧠 **Intelligent Automation (IA)**

- **How it works:** Combines RPA with AI technologies (e.g., OCR, ML) to handle semi-structured or unstructured data and make decisions.
- **Best for:** Automating workflows that involve judgment – like document classification, chatbot conversations, or exception handling.
- **Capabilities:** Adds cognitive abilities to RPA – systems can extract insights, make basic predictions, and trigger actions.
- **Analogy:** A smarter assistant that can read, understand, and act – but still within a relatively bounded scope.

🚀 **Agentic AI (Autonomous Agents)**

- **How it works:** Uses large language models (LLMs) with memory, reasoning, and tools to autonomously make decisions and take actions.
- **Best for:** Complex, dynamic tasks such as customer sentiment analysis, strategic planning, or autonomous research.
- **Capabilities:** Learns, reasons, adapts, and interacts with external tools or APIs. Operates in multi-step, iterative loops until goals are achieved.
- **Analogy:** A software agent that thinks, learns, and acts independently to fulfill a task – like an autonomous virtual employee.




| Automation Type | Intelligence Level | Task Complexity | Human Oversight |
|---|---|---|---|
| **RPA** | Rule-based | Low (structured) | High |
| **Intelligent Automation** | Context-aware + ML | Medium (semi-structured) | Moderate |
| **Agentic AI** | Reasoning + Autonomy | High (dynamic/complex) | Low (self-directed) |

# Real World Examples

**AI agents in engineering : Replit.AI, cursor.AI**

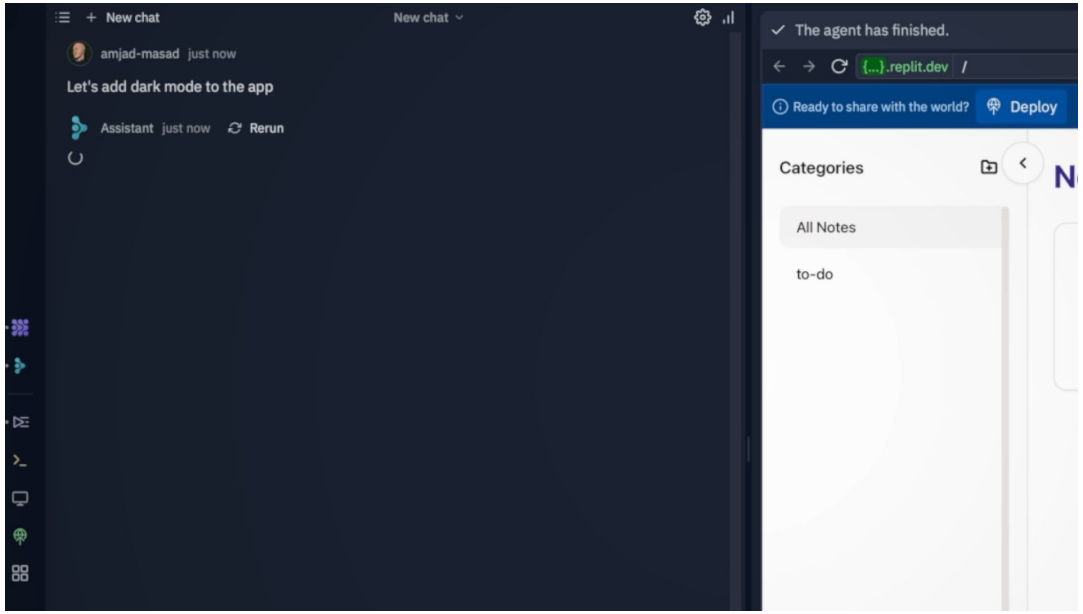
Help Engineering teams move faster and work smarter by automating common engineering tasks, optimizing resource allocation based on real-time project demands and team capabilities, identifying issues before they escalate, and streamlining time-consuming processes.

**AI agents for Business Intelligence: Zoho Zia**

AI-powered assistant for business. Helps in collecting customer data, writing a document, or just looking for some sales numbers, Zia works hard to make job easy.

**AI agents for Task efficiency: Microsoft Copilot**

AI-powered assistant for tools efficiency and daily task assistance. Microsoft copilot works alongside the user, embedded in the Microsoft 365 apps that we use every day — Word, Excel, PowerPoint, Outlook, Teams and more — to unleash creativity, unlock productivity and uplevel skills.



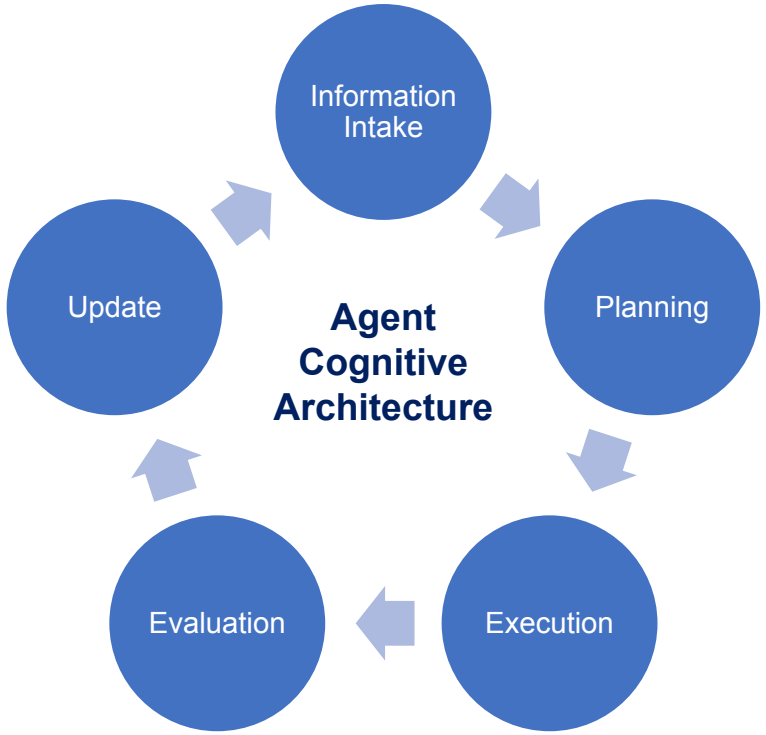

**AI Agents in IT**

empowers organizations to become more proactive and efficient, helping to resolve many common IT issues before they escalate, which can relieve a significant workload off IT's shoulders.

# Core Concepts of Agentic AI

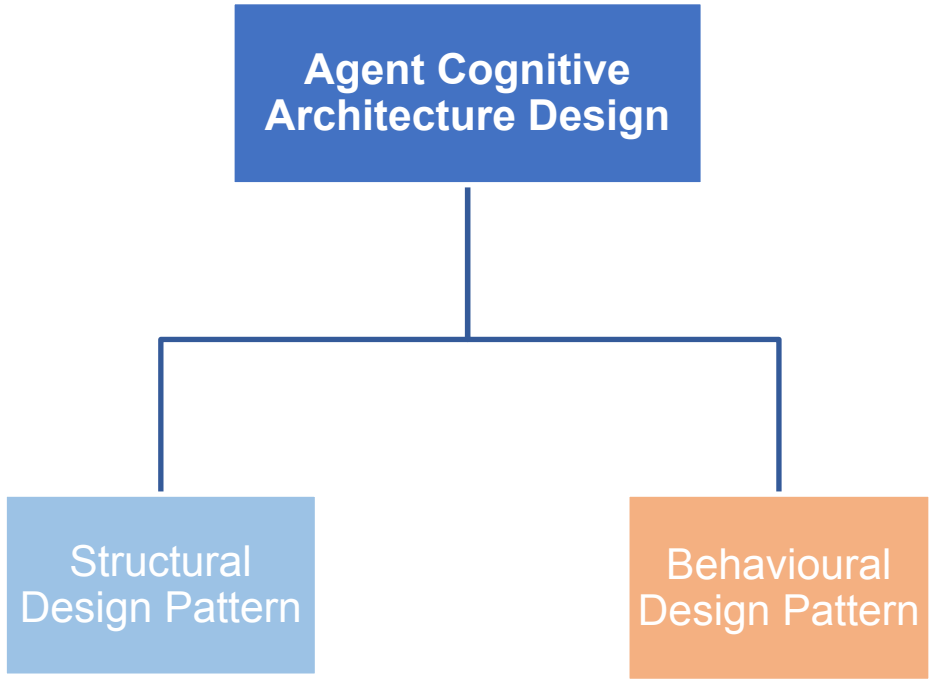
# Key Components of Agentic AI:

Autonomy, adaptability, and goal-oriented behaviour.



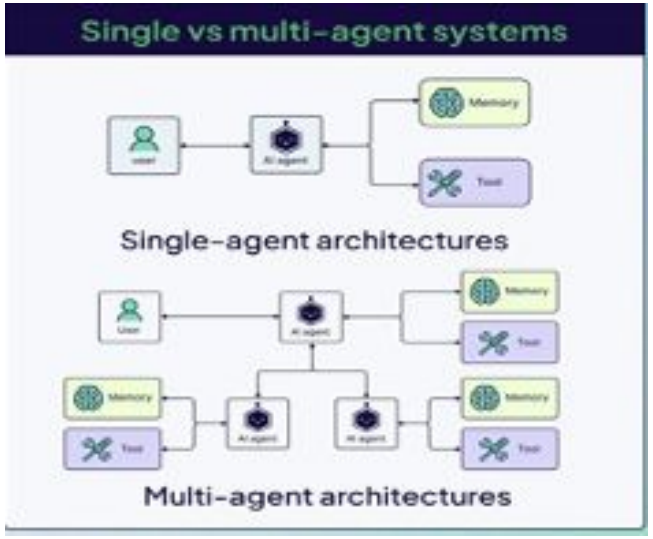

- The cyclical process of information intake, planning, executing, and adjusting describes a cognitive architecture that is unique to each Agentic system.
- At the core of agent cognitive architectures lies the orchestration layer, responsible for maintaining memory, state, reasoning and planning

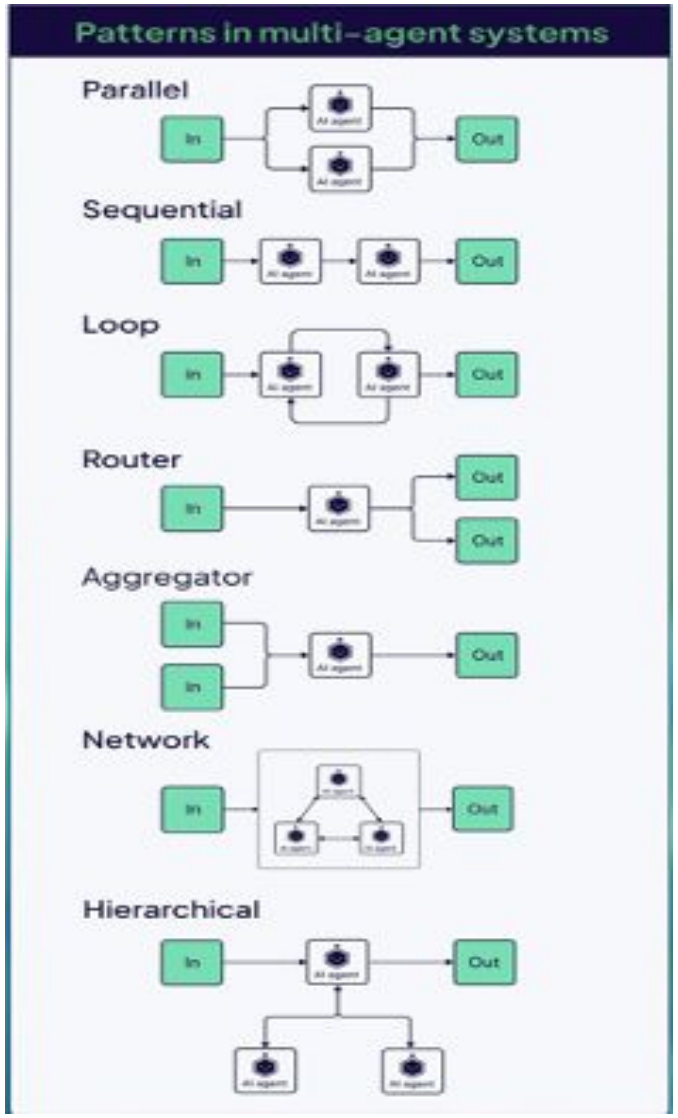
## Agent Cognitive Architecture Design

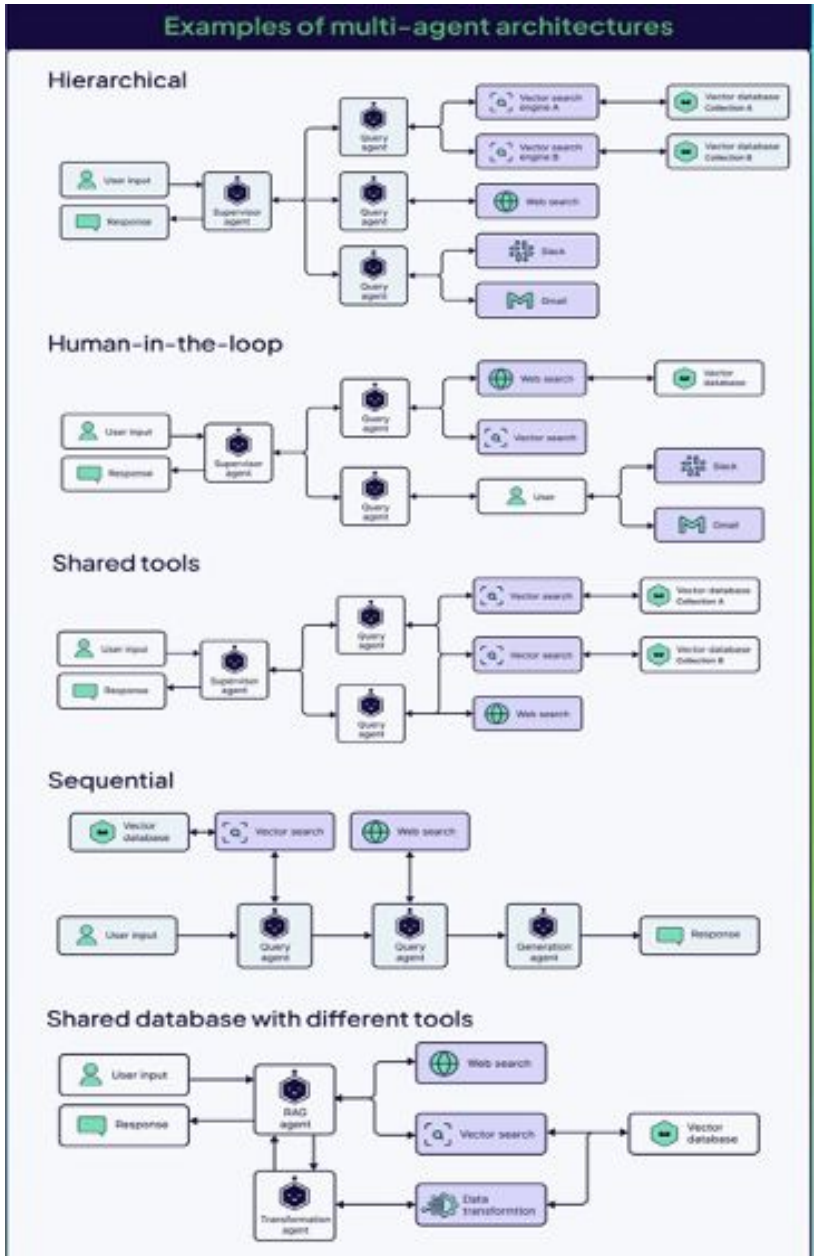


# Architecture of AI Copilots

## Agent Cognitive Architecture Design-Structural Design Pattern

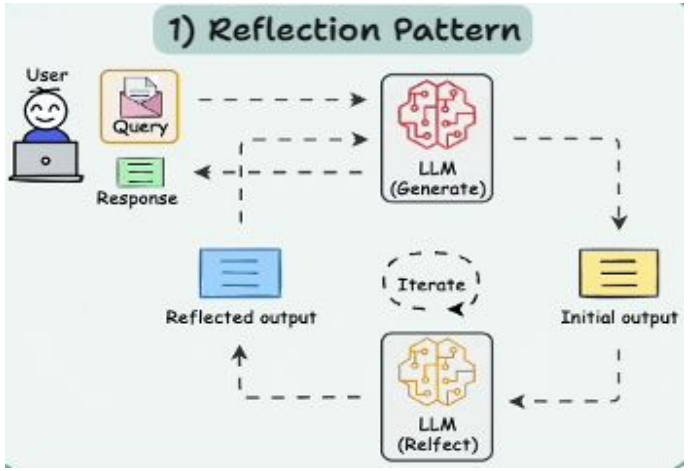


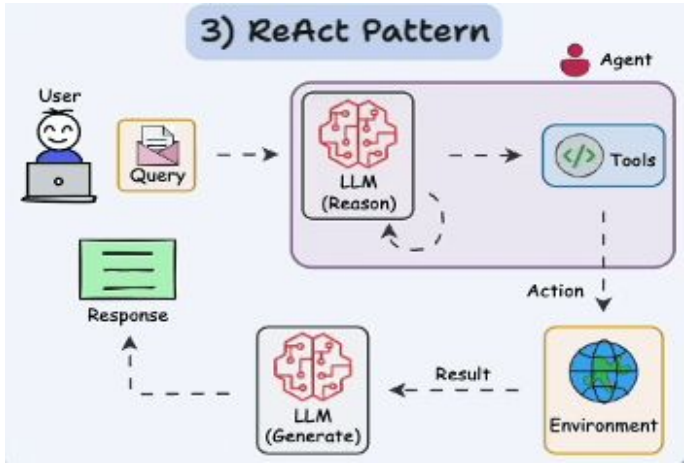



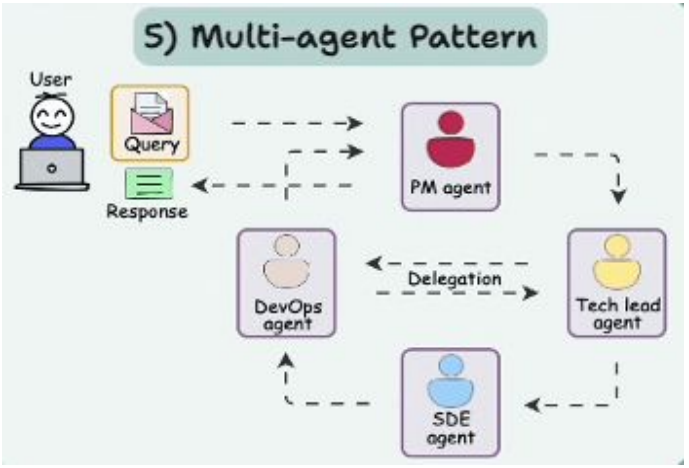



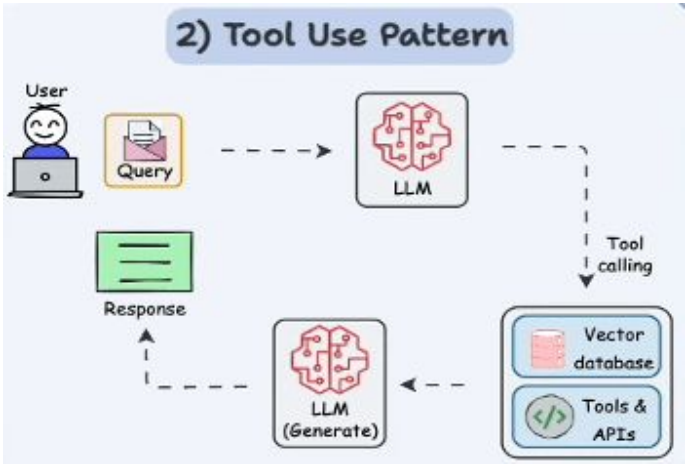
# Architecture of AI Copilots

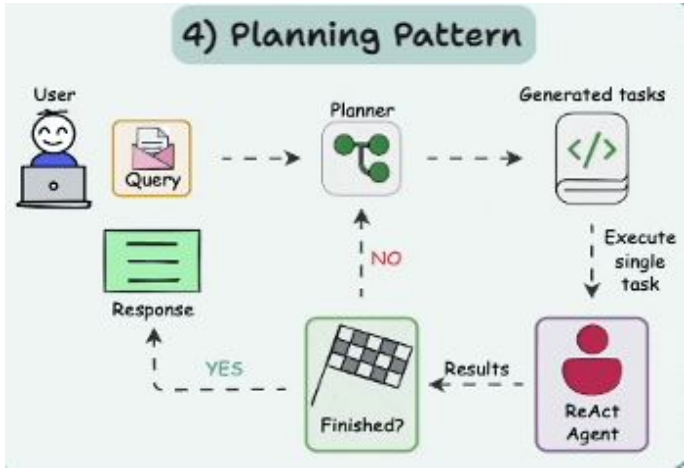
## Agent Cognitive Architecture Design - Behavioural Design Pattern



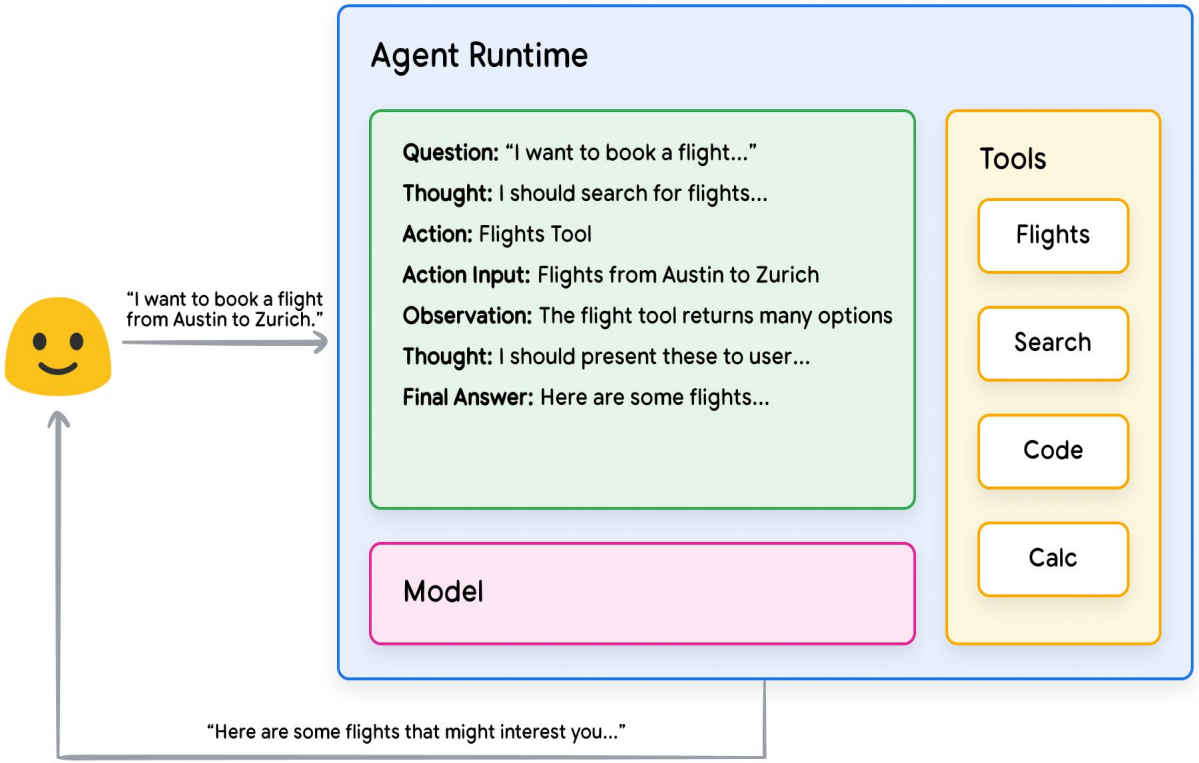



# Agent Cognitive Architecture Design : ReACT Design Pattern Example



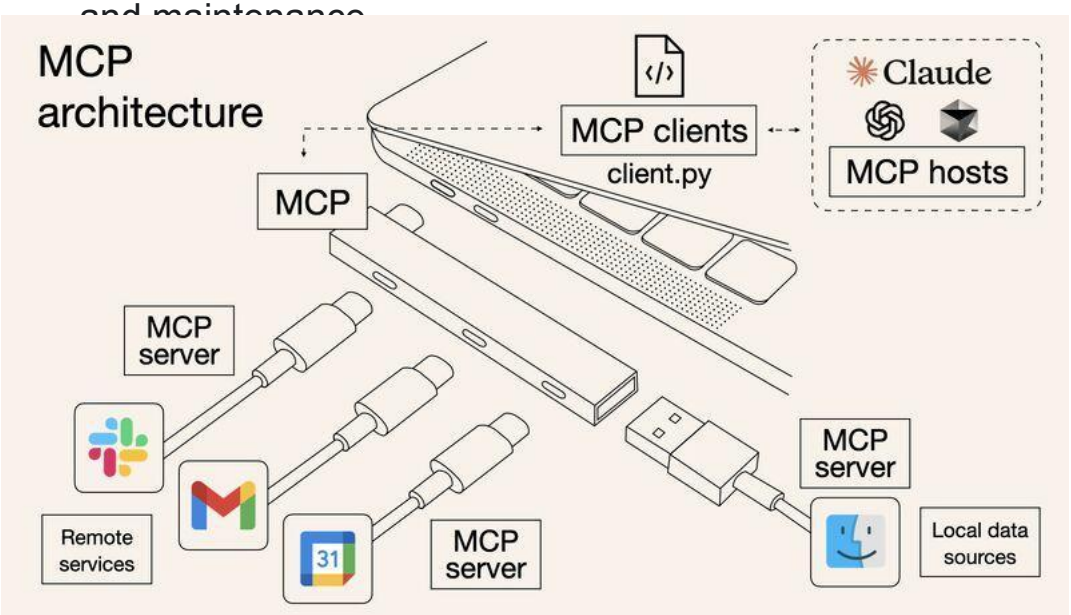

Consider an agent that is programmed to use the *ReAct* framework to choose the correct actions and tools for the user query. The sequence of events might go something like this:

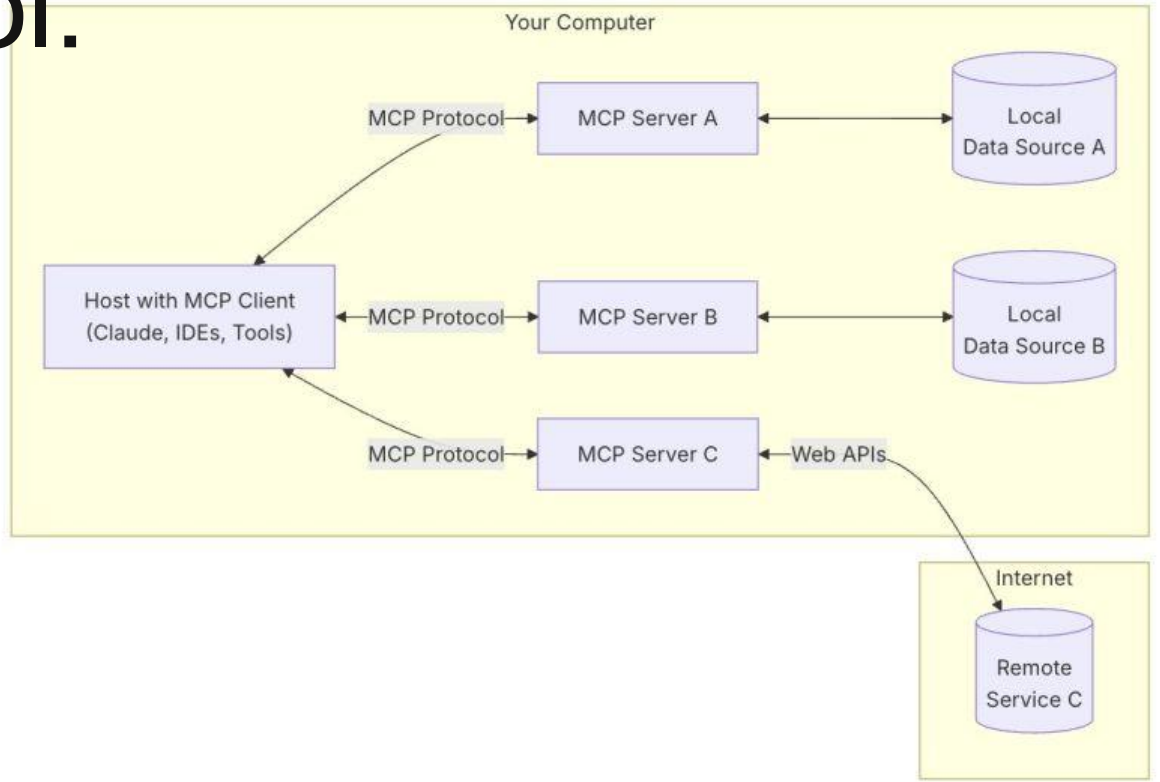
1. User sends query to the agent
2. Agent begins the ReAct sequence
3. The agent provides a prompt to the model, asking it to generate one of the next ReAct steps and its corresponding output:
    a. Question: The input question from the user query, provided with the prompt
    b. Thought: The model's thoughts about what it should do next
    c. Action: The model's decision on what action to take next
        - This is where tool choice can occur
        - For example, an action could be one of [Flights, Search, Code, None], where the first 3 represent a known tool that the model can choose, and the last represents "no tool choice"
    d. Action input: The model's decision on what inputs to provide to the tool (if any)
    e. Observation: The result of the action / action input sequence
        - This thought / action / action input / observation could repeat *N-times* as needed
    f. Final answer: The model's final answer to provide to the original user query
4. The ReAct loop concludes and a final answer is provided back to the user

# Model Context Protocol:

## Integration with APIs and external tools

- The Model Context Protocol is an open standard that enables developers to build secure, two-way connections between their data sources and AI-powered tools. The architecture is straightforward: developers can either expose their data through MCP servers or build AI applications (MCP clients) that connect to these servers.
- Think of MCP like a USB-C port but for AI agents: it offers a uniform method for connecting AI systems to various tools and data sources.
- Traditionally, connecting an AI system to external tools involves integrating multiple APIs. Each API integration means separate code, documentation, authentication methods, error handling, and maintenance.




| Feature | MCP | Traditional API |
|---|---|---|
| **Integration Effort** | Single, standardized integration | Separate integration per API |
| **Real-Time Communication** | ✅ Yes | ❌ No |
| **Dynamic Discovery** | ✅ Yes | ❌ No |
| **Scalability** | Easy (plug-and-play) | Requires additional integrations |
| **Security & Control** | Consistent across tools | Varies by API |