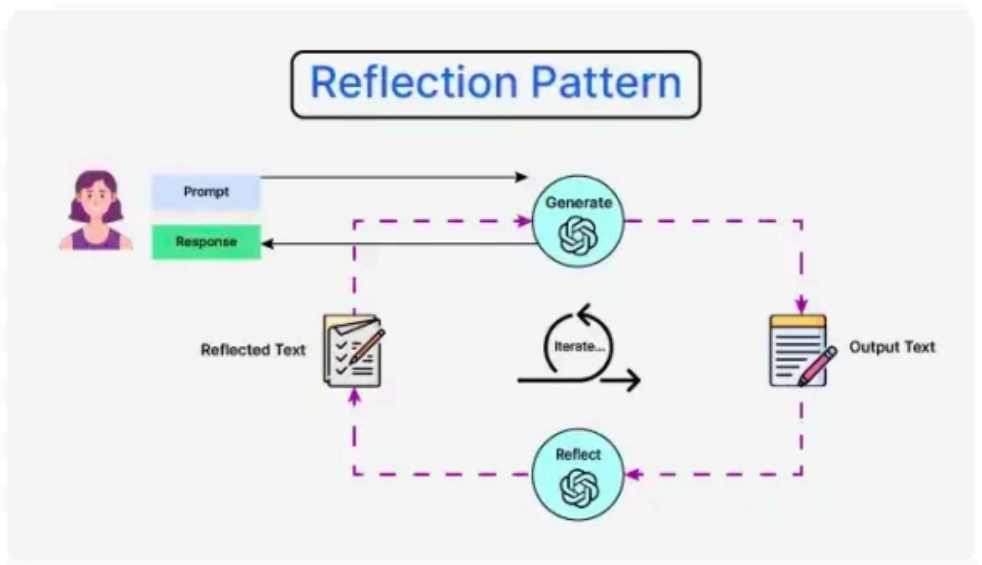
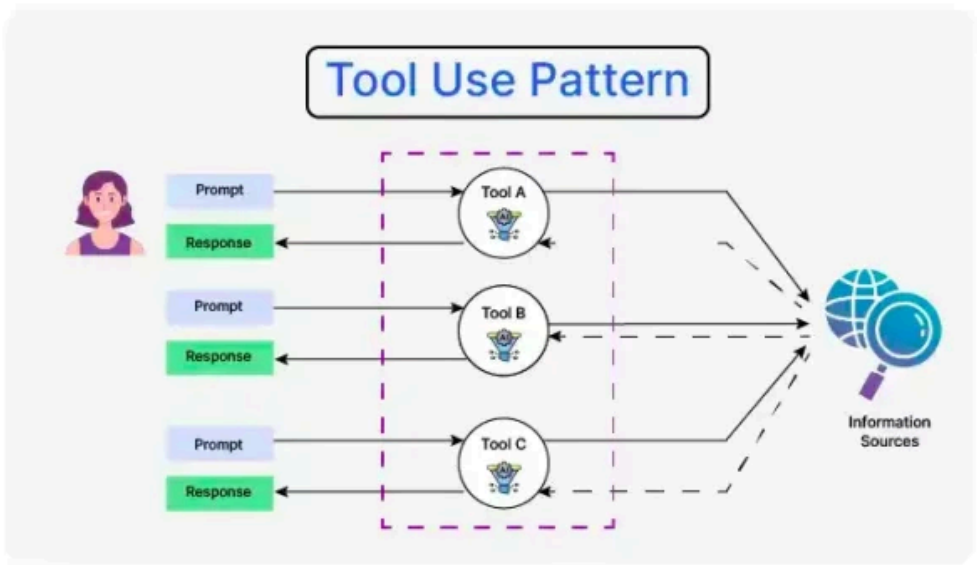
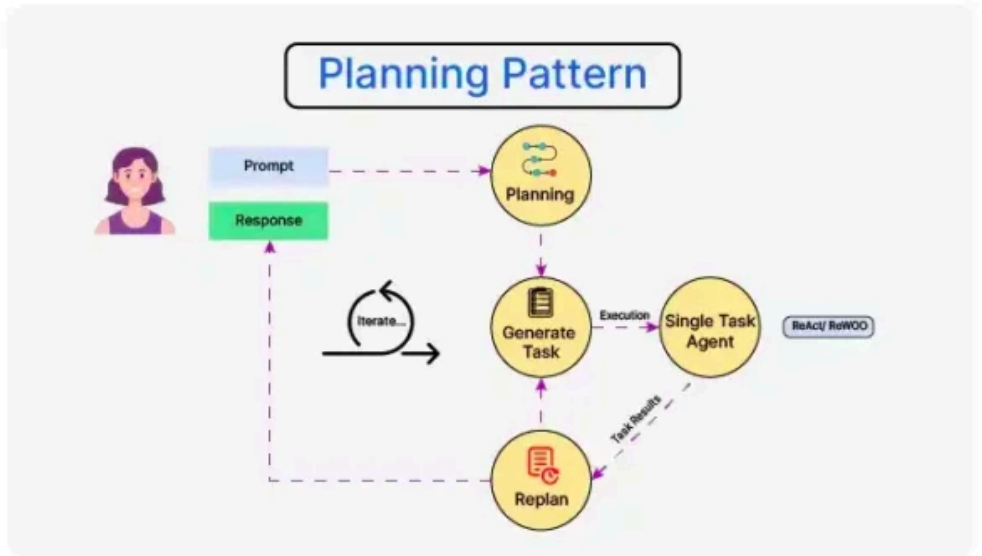
Analytics Vidhya

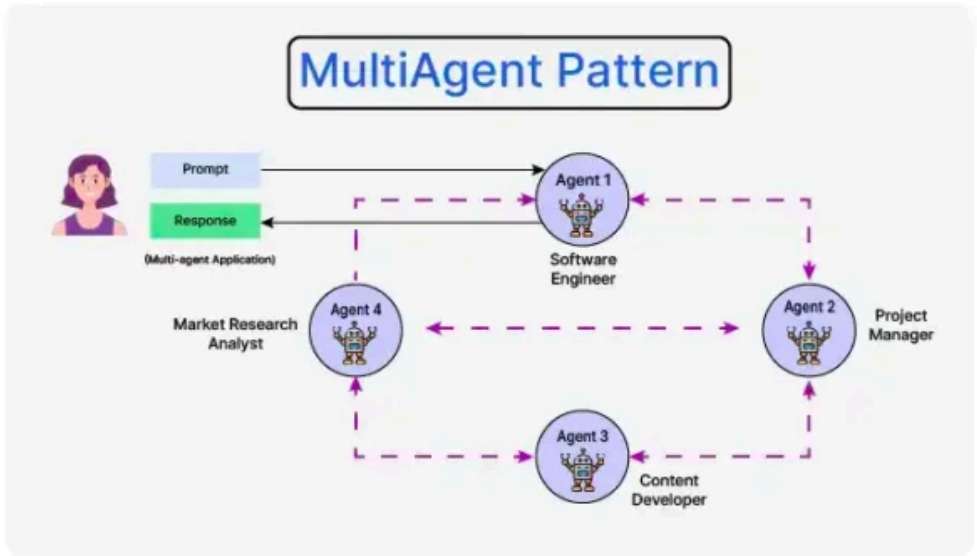
8 Things to Keep in Mind while Building AI Agents

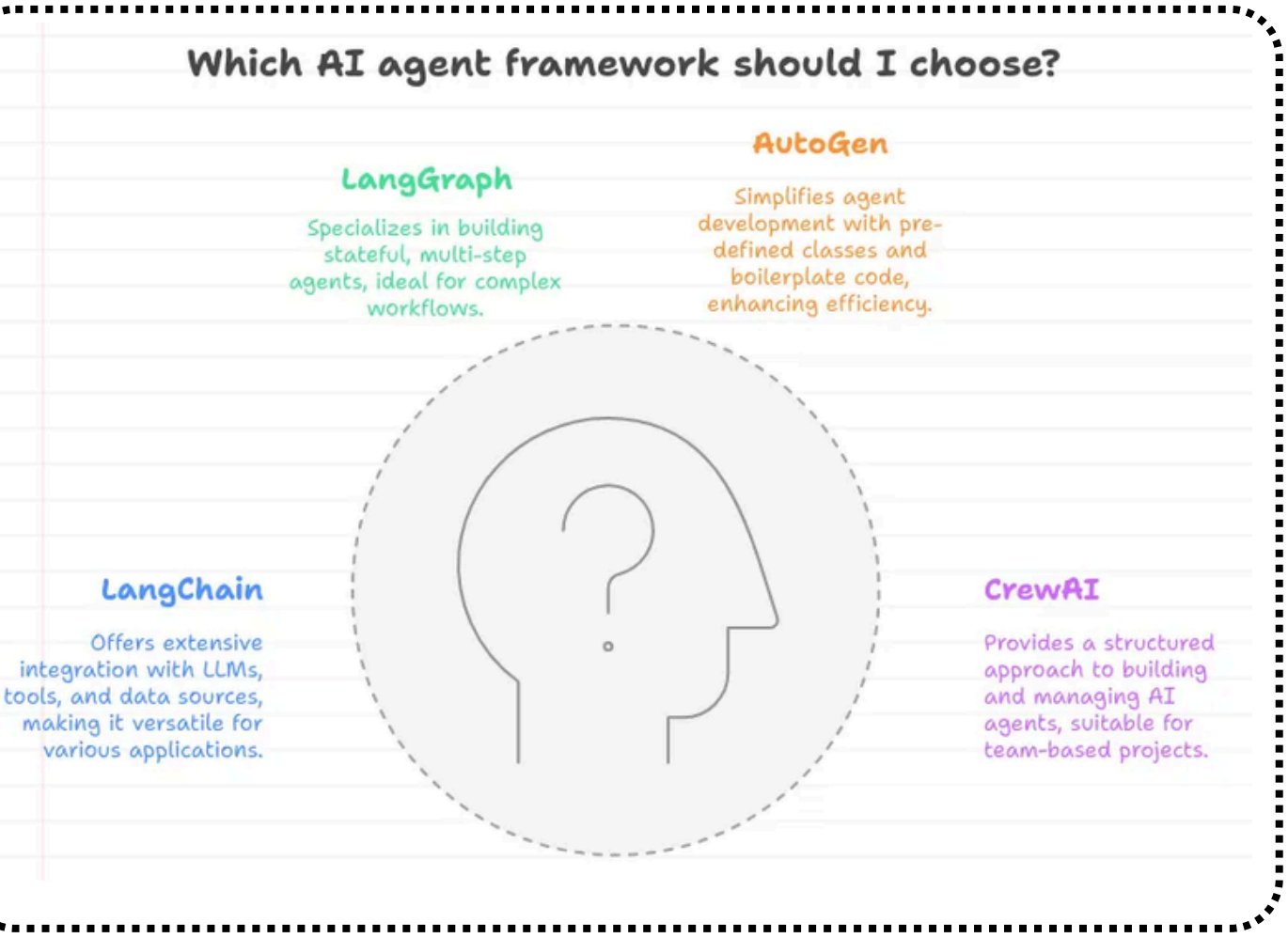
Agentic Design Patterns

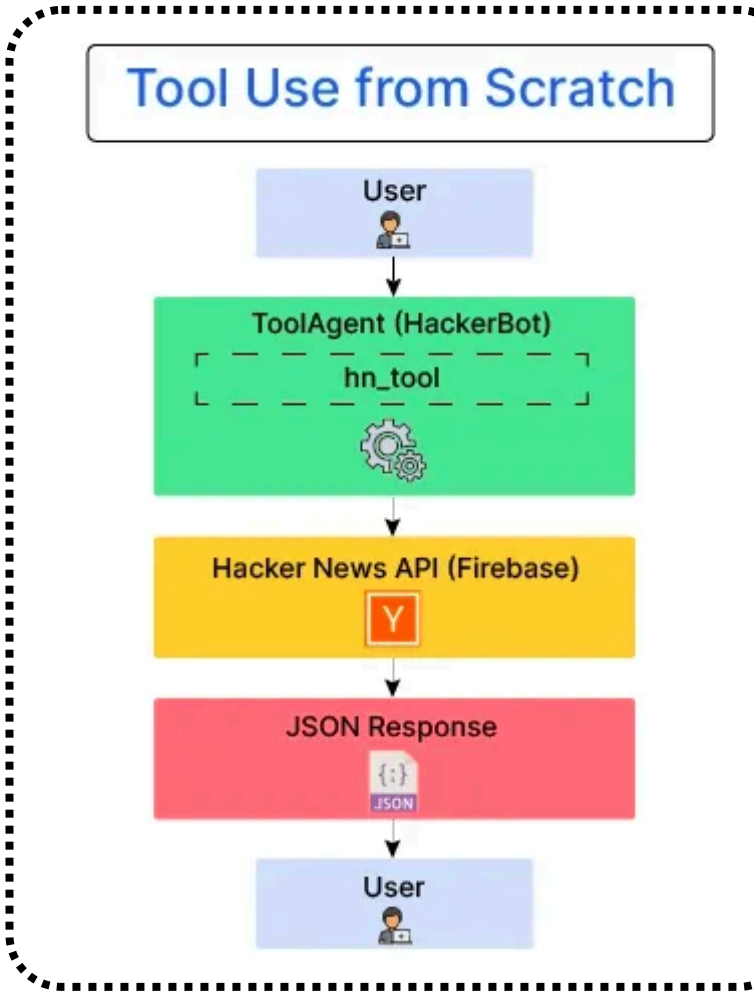






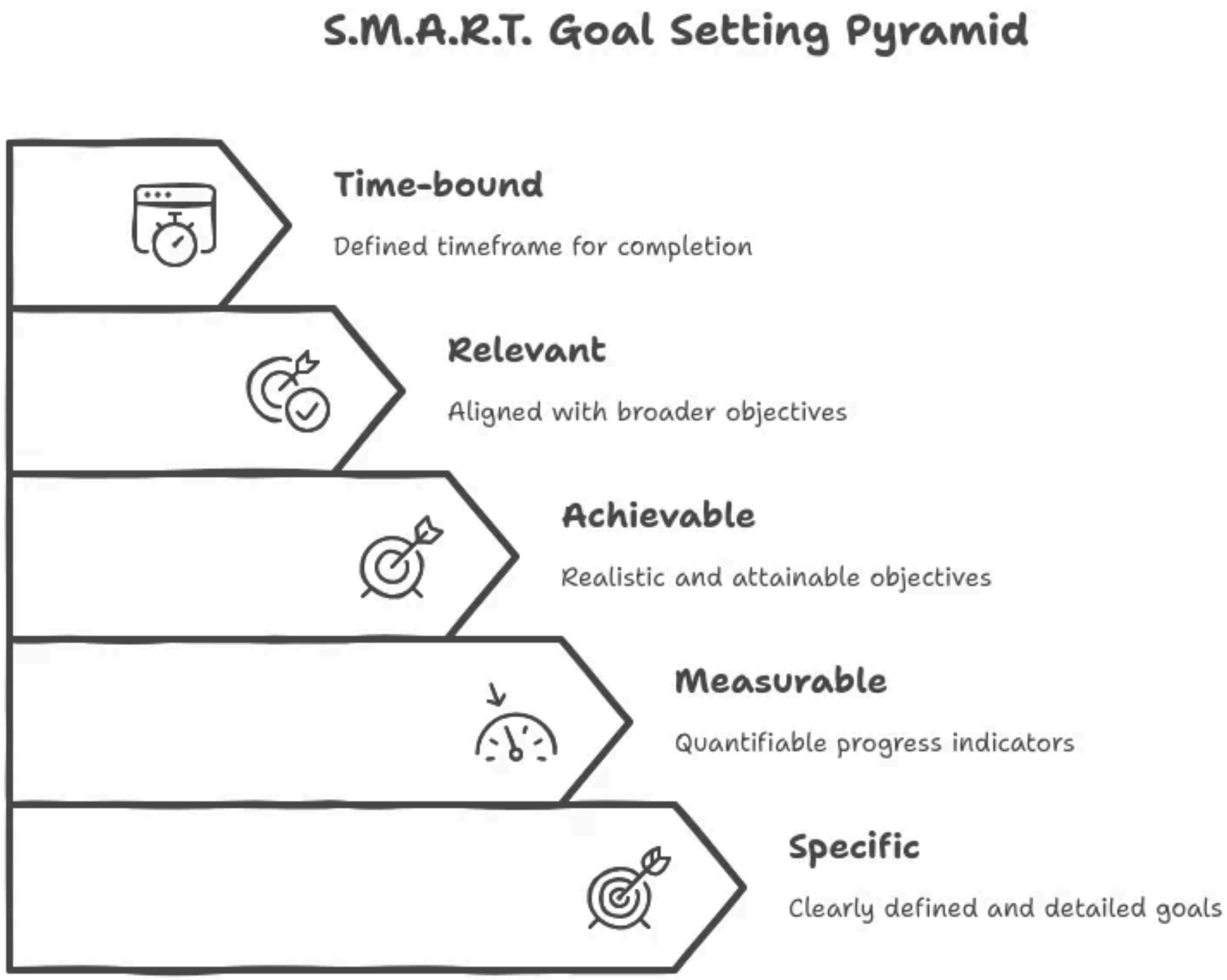






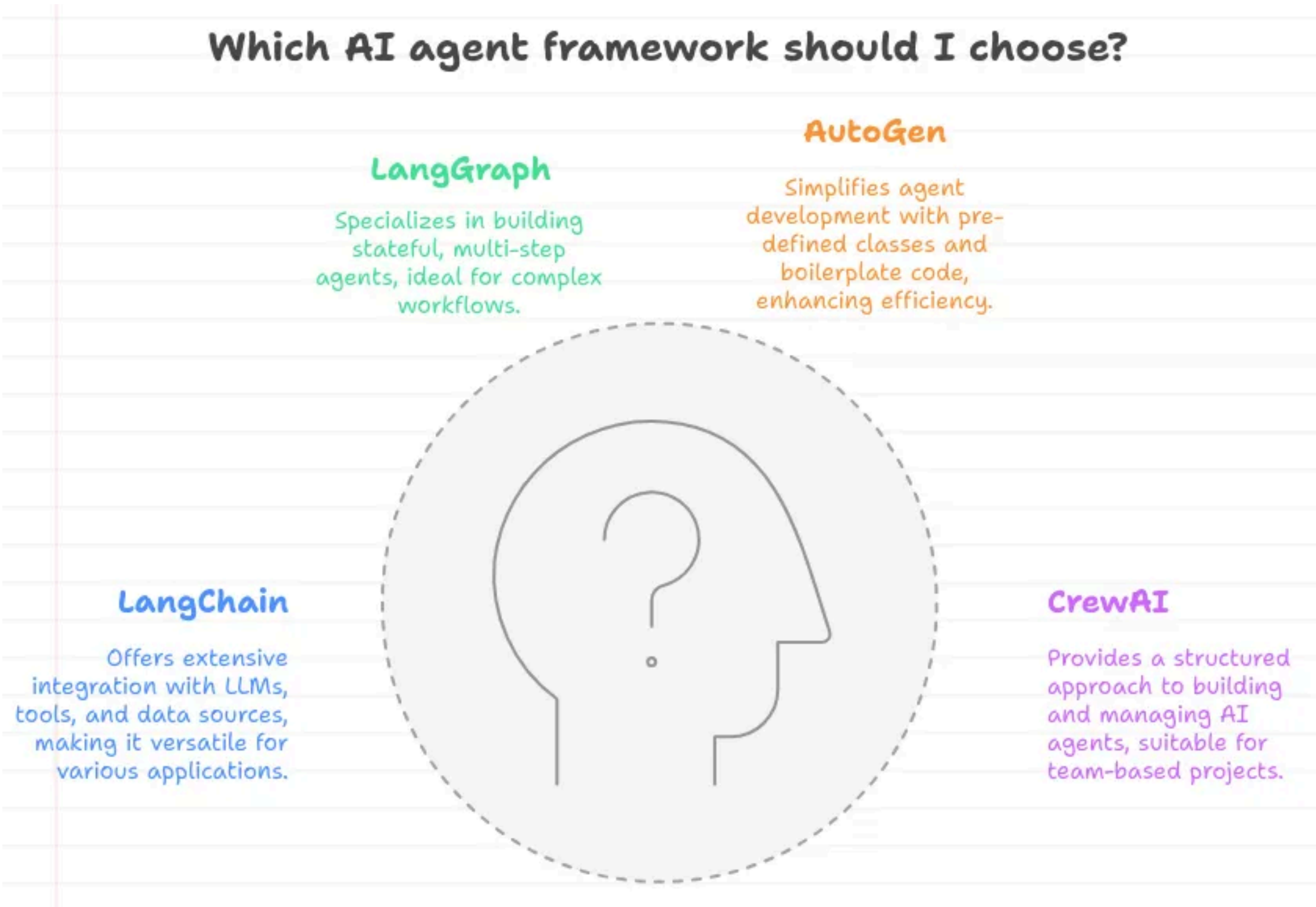
Define the Agent's Goal Clearly

The foundation of any successful AI agent is a clearly defined goal. People often create objectives that are vague and not detailed, which leads to very generic results and hallucinations. Think of it like ordering a human to do a task with an unclear objective; for sure, he will mess up. So, the agent needs to know in detail which task it needs to perform and how to perform it. Otherwise, it cannot work efficiently. For building AI agents that deliver, being specific is mandatory.



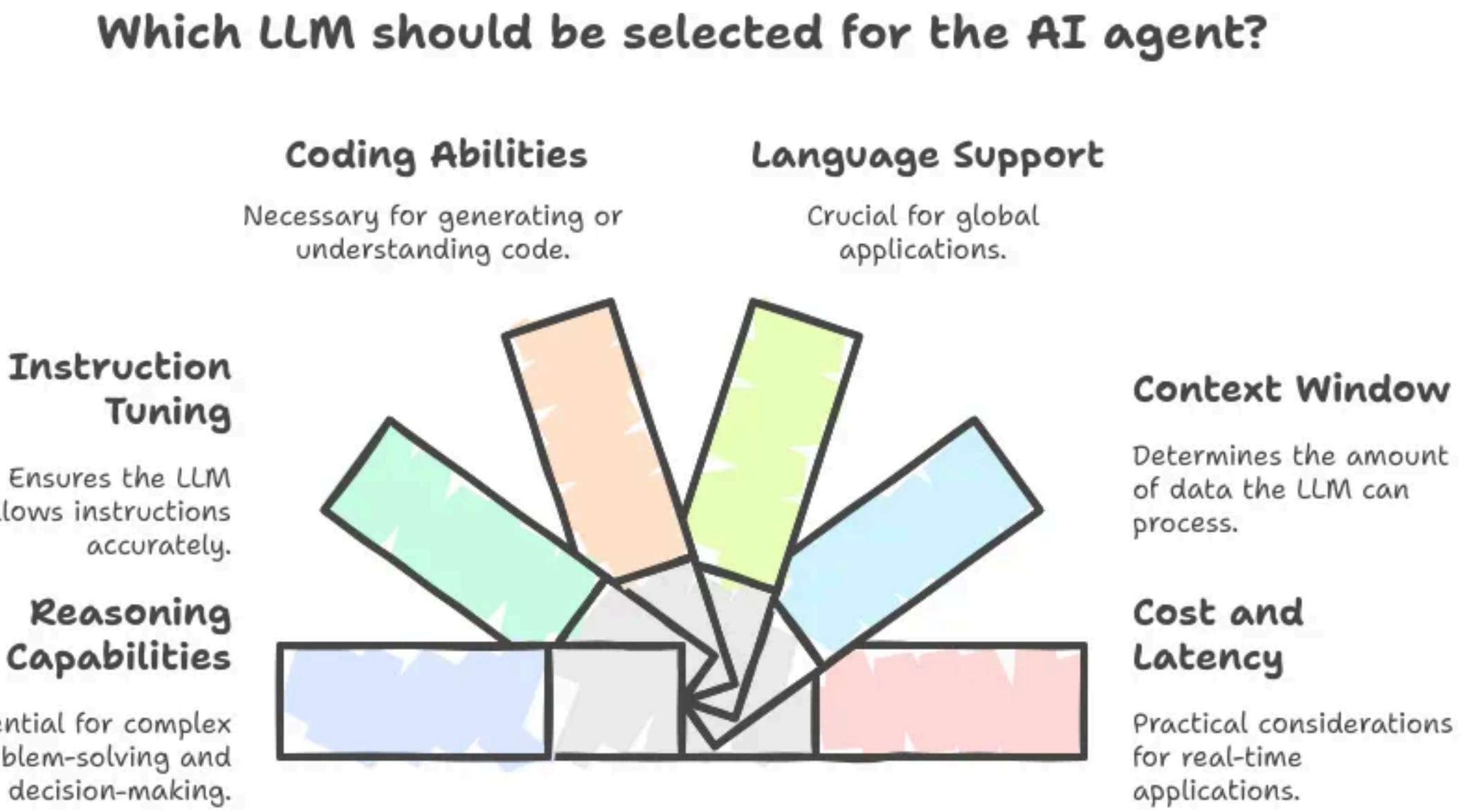
Choose the Right Framework

Building the AI agents from scratch can be a complex task. Luckily, several frameworks simplify this process. Making AI agents using them feels like a cakewalk. LangChain, LangGraph, AutoGen, or CrewAI provide a very structured way to build, deploy as well as manage AI agents. They contain pre-defined classes, tools, as well as boilerplate code that speeds up the development in a very efficient manner.



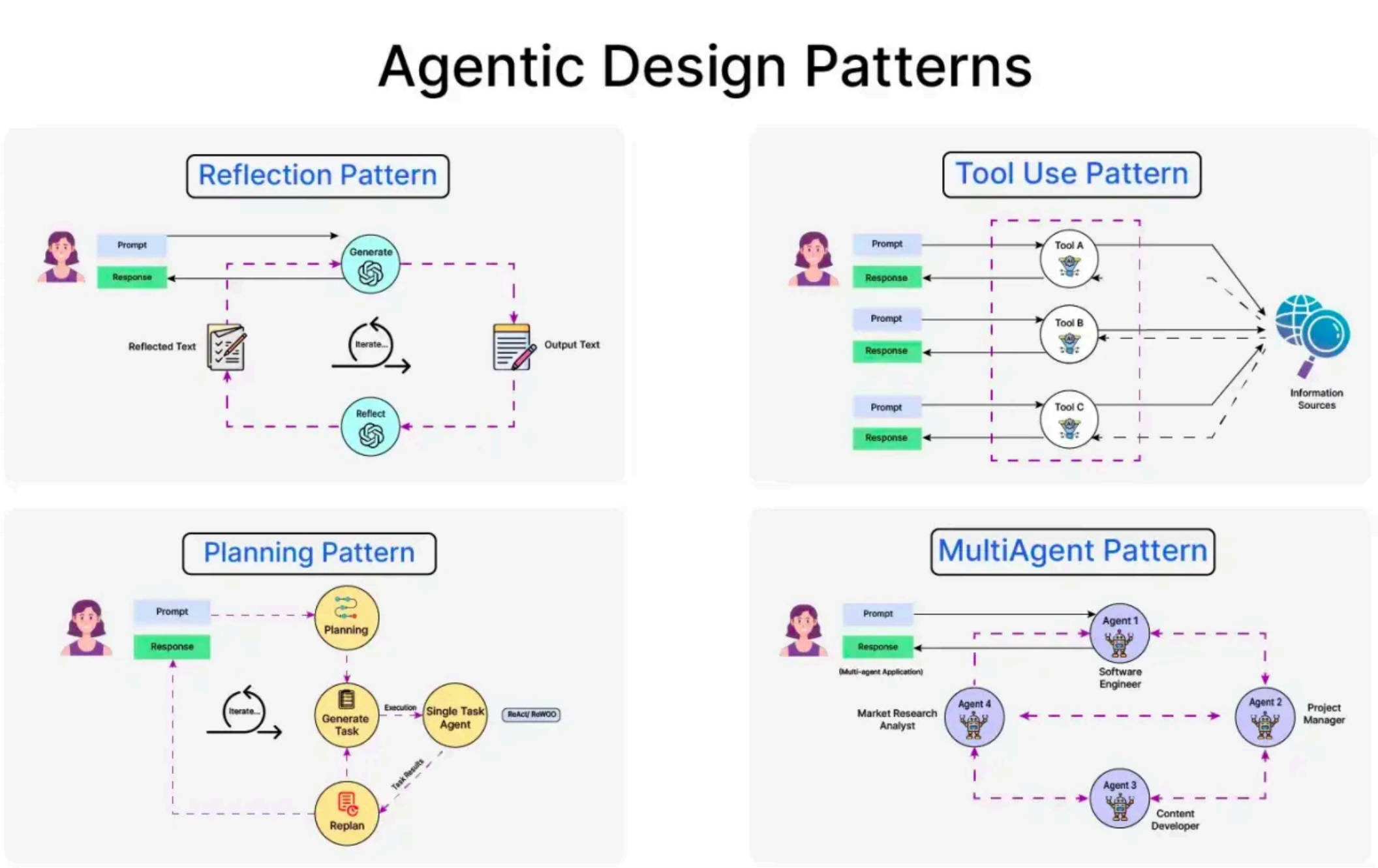
Select the Appropriate LLM

Selecting the right Large Language Model (LLM) is a crucial step. LLM acts as the “brain” of your AI agent. The LLM’s capabilities directly affect your agent’s performance in the production environment. It determines how intelligent and smart your agent will be. In 2025, the market offers several LLMs, each having different advantages and strengths. Top LLMs in the market include OpenAI’s GPT series, Meta’s Llama models, Anthropic’s Claude, or Google’s Gemini.



Choose the Right Agent Architecture

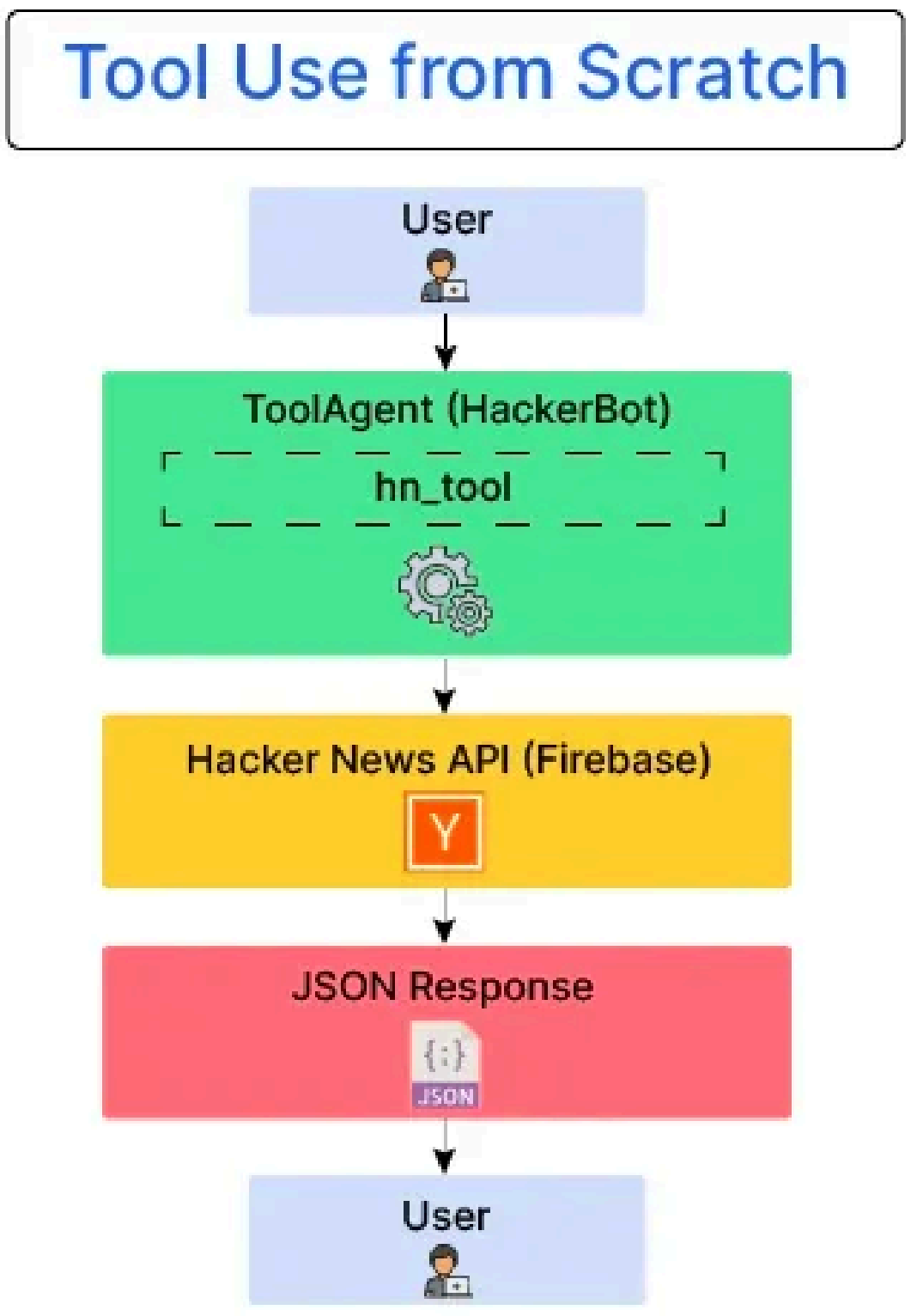
The design pattern and specific architecture of the AI agent are critical. This defines how a particular agent processes information, executes its tasks, and makes desired decisions accordingly. There are various architectures suited to different types of problems.



A popularly used architecture is ReAct, which means first reason and then act. Agents using this architecture first reason over the user's query and then decide the next best action based on their reasoning, and then execute it.

Tool Integration

LLMs on their own are very powerful and efficient, but they operate in a very isolated environment, unaware of the real world. Tool integration provides these agents the ability to interact with the real world and perform meaningful actions. Tools allow the AI agents to access the external information or trigger actions in other systems. This is like providing your LLM superpowers so that it can do several tasks on its own, which often requires human intervention.

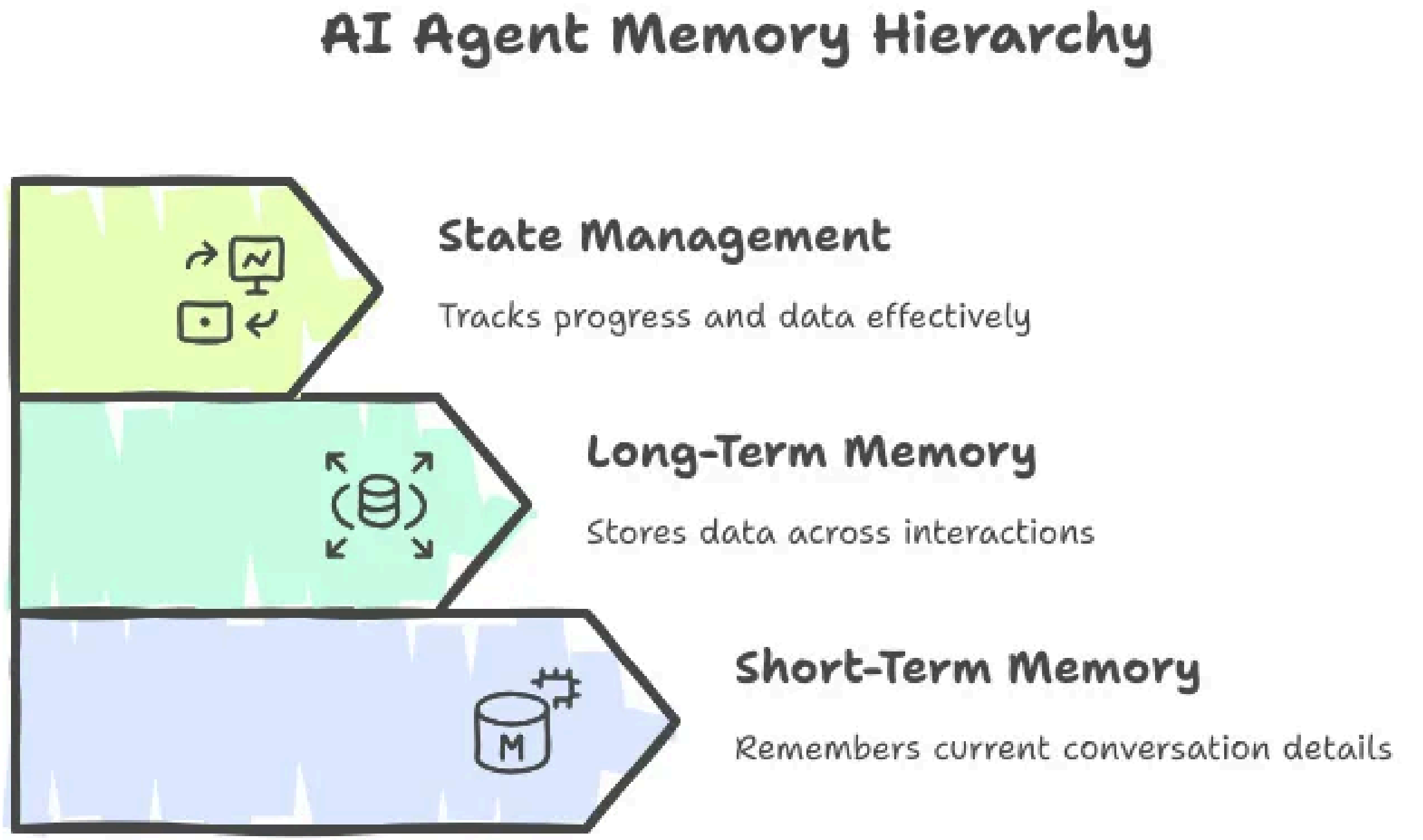


Examples of tools integration include math tools, APIs for accessing external data like weather updates or stock market prices, and triggering some event, like sending emails. Agents need these tools to query the databases and perform web searches for updated information.

Code execution tools allow agents to execute the scripts. These tools must be reliable for use in real life. AI agents decide which tool to use while running.

Memory and State Management

AI agents can be truly useful for long-running tasks or conversations. It needs memory for this. By utilizing the memory, the agent can recall the past conversations and maintain the context of the conversation while answering questions. Without memory, each interaction is new, and it limits the agent's ability to learn or build from previous interactions. State management is key to building AI agents that feel coherent.



Prompt Engineering and Few-Shot Examples

One of the primary ways to communicate with AI agents and their “brain” LLMs is through Prompts. The accuracy as well as the behaviour of the agent is heavily dependent on the quality of prompts you pass to the agent.

Defining clear, non-ambiguous prompts is a necessary skill. Effective prompt engineering is vital when building AI agents.

Overall personality of the agent, its objectives, and behavior are extensively defined by the System prompt. For achieving a specific task, the agent must be provided with a few-shot example with the prompt so that the agent can understand the intention of the question and the expected format of the answer. This, in turn, can significantly improve the performance.

Providing LLMs with certain examples can help them understand the task very well. These careful instruction helps align the agent with your expectations.

Evaluation and Feedback Loop

Once you are done building your AI agent, you need to evaluate how it's performing. Hence, continuous evaluation is essential. You have to set some metrics from the very first step to carefully evaluate the performance. These metrics should be aligned with the agent's defined goals. Testing should be performed to evaluate the agent. This can include various end-to-end tests for evaluating the agent's behaviour in different scenarios. Performing unit tests on individual tools is an important aspect of Automated testing. However, for highly complex tasks with complex reasoning, human evaluation is mandatory. Human feedback can identify the unexpected failures in the behaviour of the AI agent.