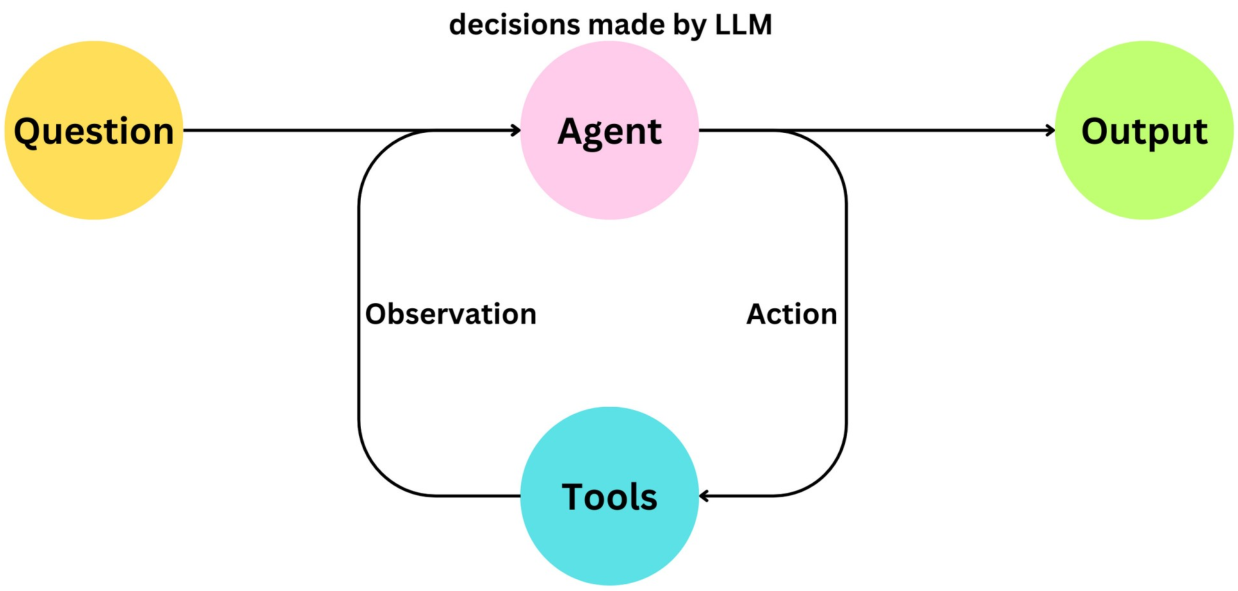
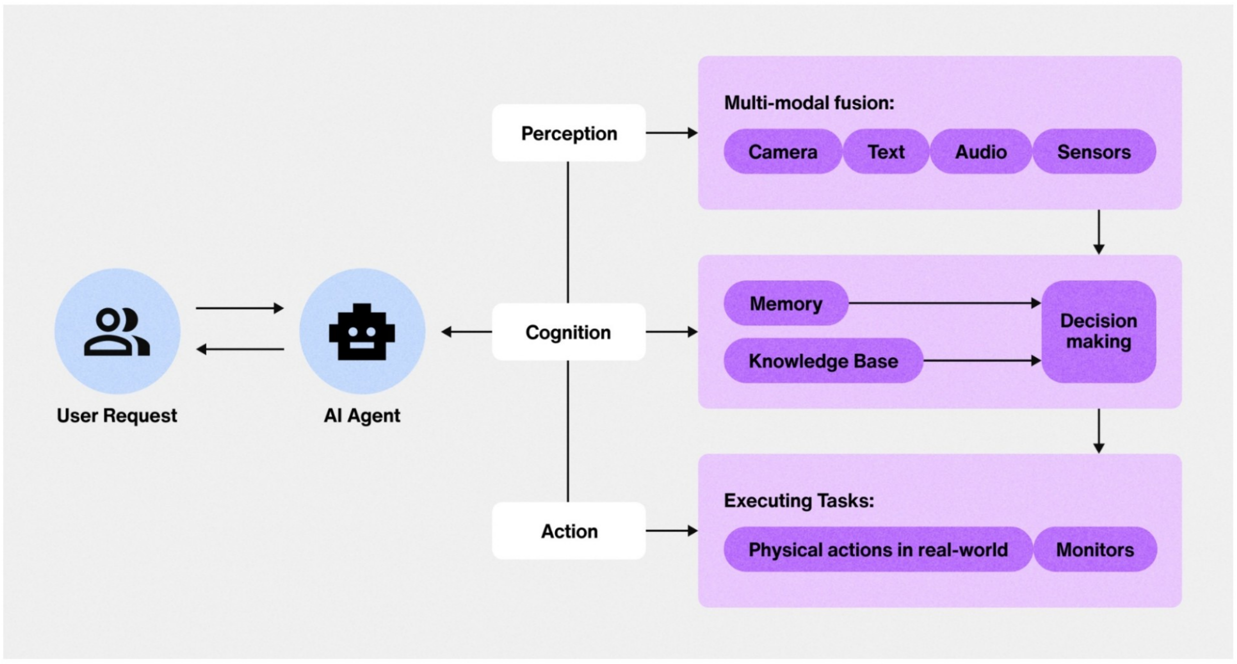
AI Agents

Agents are AI systems that can dynamically choose which tools to use based on the task at hand. They can reason about problems, decide what actions to take and use external tools to gather information or perform tasks.



- **Action**: The LLM decides what action to take based on the user's request and available tools.
- **Observation**: The agent executes the chosen action using a tool and observes the outcome.
- **Thought**: The LLM processes the observation, potentially deciding if further actions are needed or if the task is complete.
- **Final Answer**: Once the agent determines it has achieved the desired outcome, it provides the final response

Agent can do a wide range of tasks



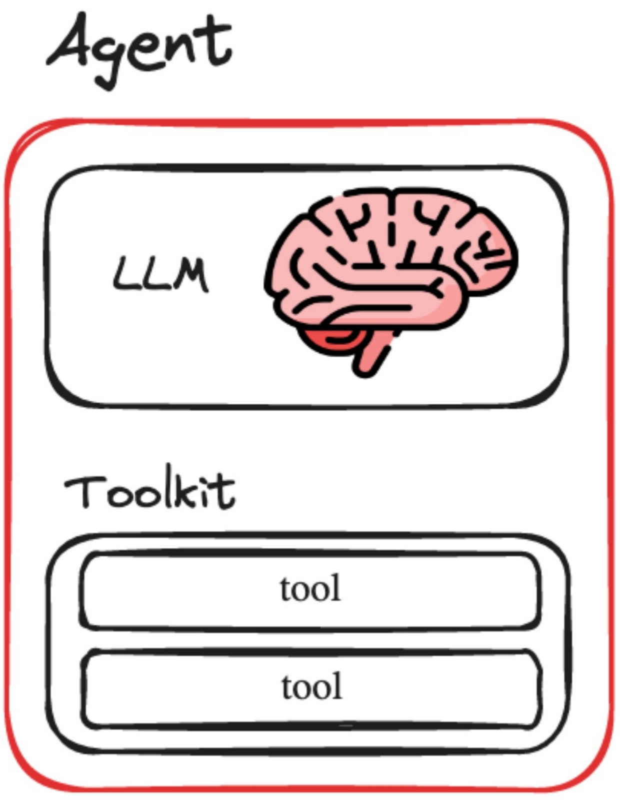
src = https://sendbird.imgix.net

```
In [4]: gemini = ChatGoogleGenerativeAI(model='gemini-2.5-flash', temperature=0.8)
```

Tools in agents

Tools are what makes an LLM an agent.

AI agents leverage tools to enhance their capabilities, allowing them to interact with the world, access information, and perform tasks beyond their core reasoning abilities. These tools can include web search, code execution, database access, and interactions with external APIs.



A simple way to think of these is to giving the LLM access to some script it can run independenlty.

List of custom tools we made in the full Notebook

In [16]:
```python
for t in tools:
    print(f'{t.name} : {t.description}\n')
```

```
Calculator : Use this for mathematical calculations with numbers, make sure input is a val
id expression like "2+4/3-(42)" and "21%7" etc.

Text Analyzer : Analyze text and provide statistics like word, sentence and character coun
t.

Date and Time : Get the current date and time. Input can be a timezone (default in UTC)

Unit Converter : convert between common units. FORMAT : source_val source_units to target_
units
```

Eg : Text Analyzer

analysizing the text and providinng statistics related to the text

In [10]:
```python
def text_analyze(text: str)-> str:
    words = text.split()
    sentences = text.split('.')
    chars = len(text)

    stats = {
        'word_count' : len(words),
        'sentence_count' : len(sentences),
        'character_count' : chars
    }
    # Not Perfect but it works for our use case

    res = ''
    for k, v in stats.items():
        res += f'\n\t{k} : {v}'
    return res
```

In [11]:
```python
text_analysis_tool = Tool(
    name = 'Text Analyzer',
    description = 'Analyze text and provide statistics like word, sentence and character
    func = text_analyze
)

tools.append(text_analysis_tool)
```

Creating Agents

At its core, a LangChain agent's structure is designed for dynamic decision-making and interaction with the world around it . It's fundamentally different from a chain, which follows a predefined sequence of actions.

Here are the major elemeents:

- **Language Model (LLM)**: This is the brain of the agent, responsible for understanding inputs, reasoning, and deciding the next action.
- **Tools**: These are external resources, APIs, or functions that the agent can use to perform specific tasks, like searching the web, performing calculations, or interacting with a database.
- **Agent Executor**: This component manages the agent's actions, executes them through the tools, and incorporates the results back into the agent's reasoning process

In [17]:
```python
from langchain.agents import AgentType, initialize_agent
```

Zero-shot React Agent (most common)

They are used when only a singular response is need i.e., no back and forth conversation occur

In [18]:
```python
react_agent = initialize_agent(
    tools = tools,
    llm = gemini,
    agent = AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose = True,
    max_iterations = 3,
    early_stopping_method = 'generate'
)
```