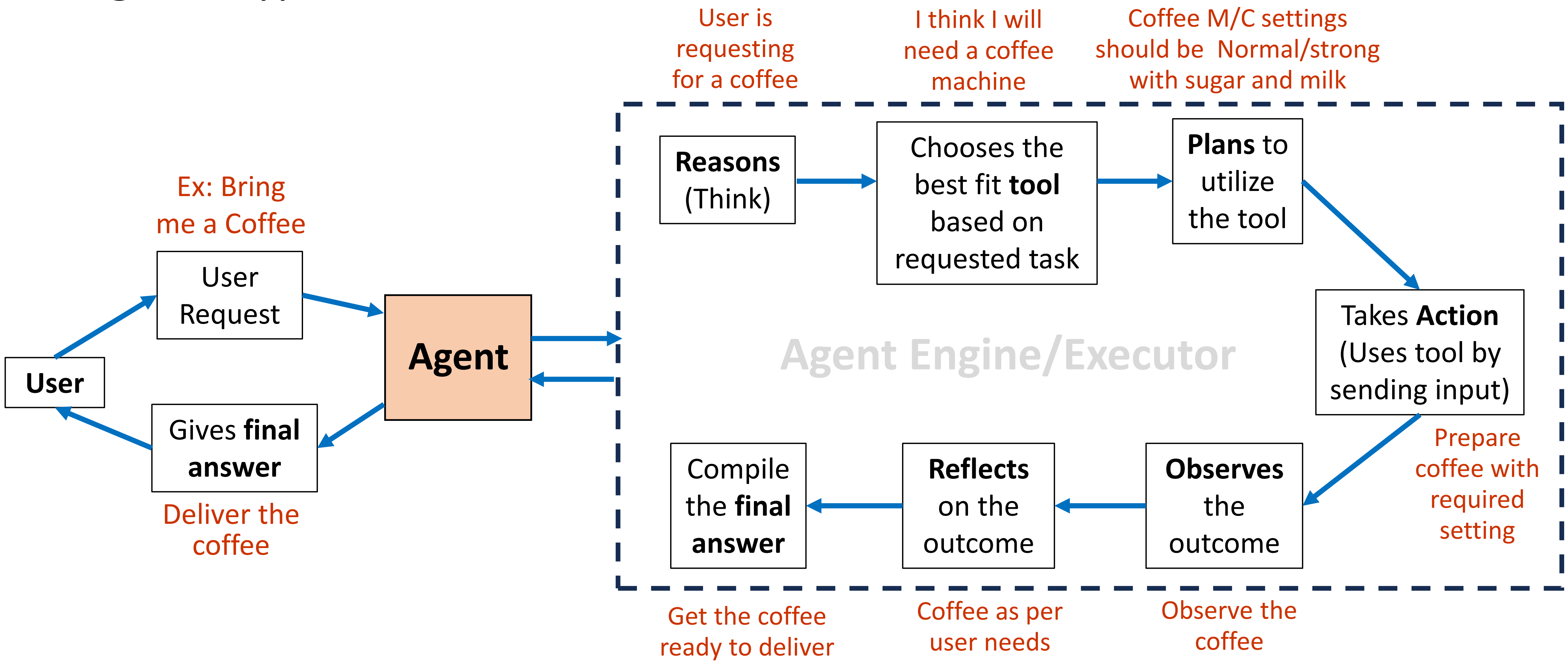
Agents & Agentic Framework

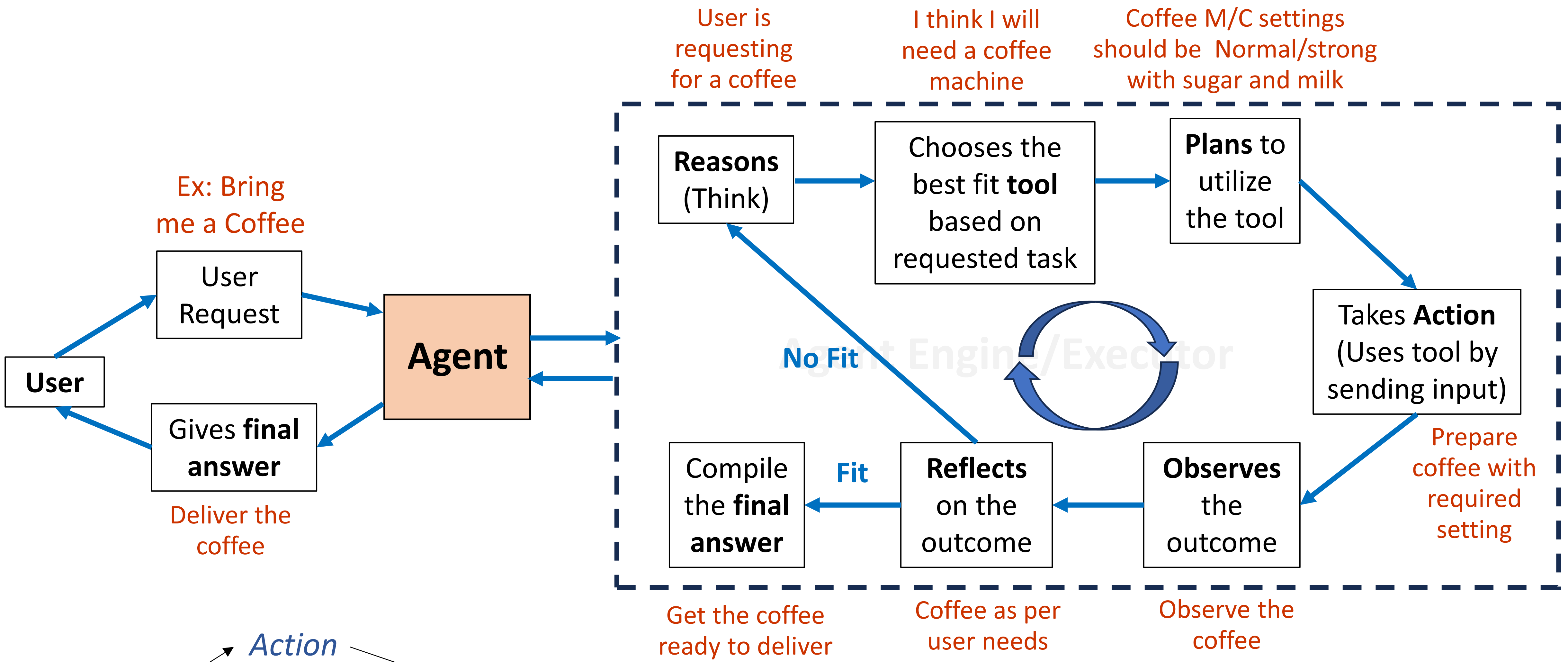
Bhanu Chander V

24th June 2025

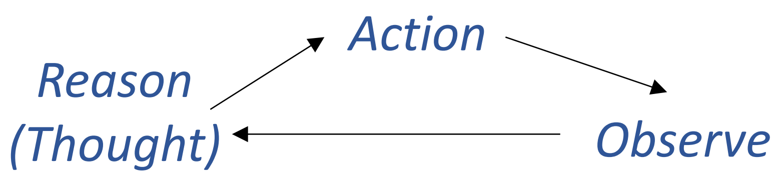
Agent - Typical Process


Ex: Bring me a Coffee
User Request
User
Agent
Gives **final answer**
Deliver the coffee
User is requesting for a coffee
I think I will need a coffee machine
Coffee M/C settings should be Normal/strong with sugar and milk
Reasons (Think)
Chooses the best fit **tool** based on requested task
Plans to utilize the tool
Agent Engine/Executor
Takes **Action** (Uses tool by sending input)
Prepare coffee with required setting
Observes the outcome
Reflects on the outcome
Compile the **final answer**
Get the coffee ready to deliver
Coffee as per user needs
Observe the coffee

Agentic Framework *is Iterative*

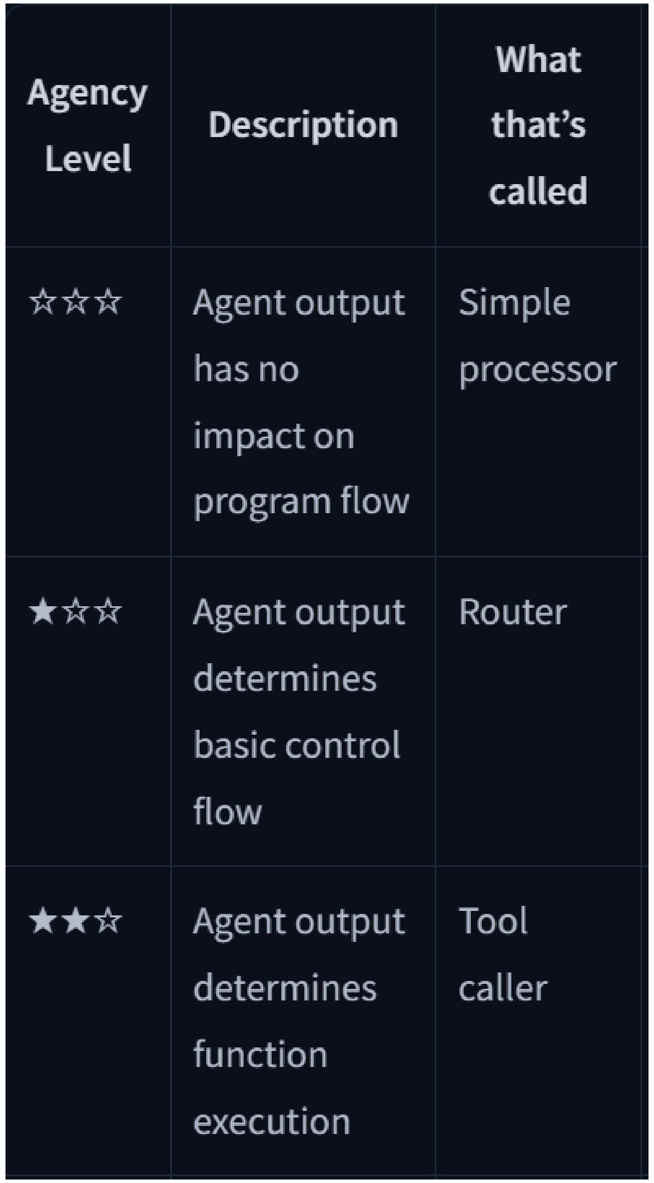
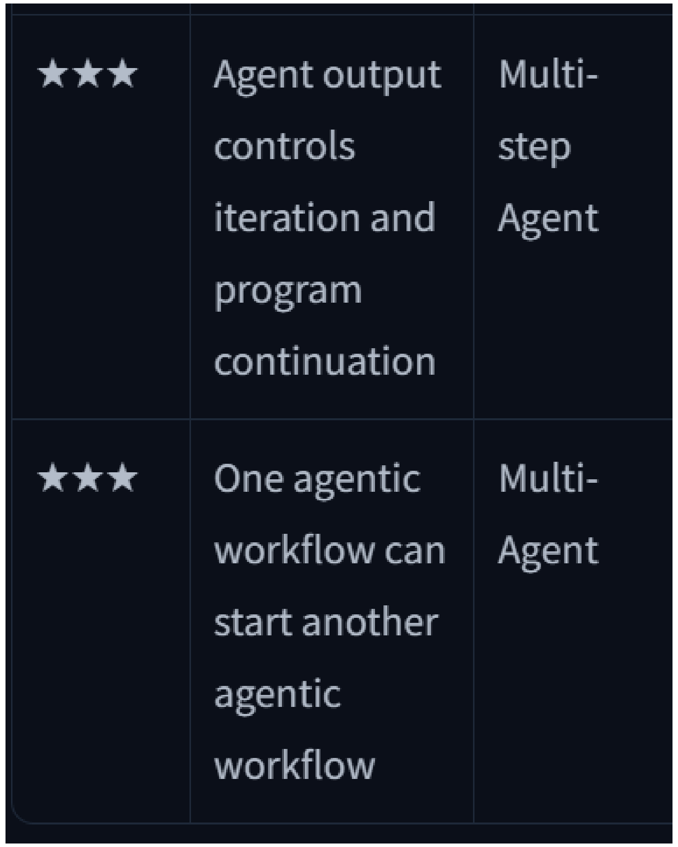


User is requesting for a coffee
I think I will need a coffee machine
Coffee M/C settings should be Normal/strong with sugar and milk
Ex: Bring me a Coffee
User Request
User
Agent
Gives final answer
Deliver the coffee
Reasons (Think)
Chooses the best fit tool based on requested task
Plans to utilize the tool
Takes Action (Uses tool by sending input)
Prepare coffee with required setting
Agent Engine/Executor
No Fit
Fit
Compile the final answer
Reflects on the outcome
Observes the outcome
Get the coffee ready to deliver
Coffee as per user needs
Observe the coffee



ReAct Approach

Agent Levels

Agency Level	Description	What that's called
☆☆☆	Agent output has no impact on program flow	Simple processor
★☆☆	Agent output determines basic control flow	Router
★★☆	Agent output determines function execution	Tool caller
★★★	Agent output controls iteration and program continuation	Multi-step Agent
★★★	One agentic workflow can start another agentic workflow	Multi-Agent

Agentic frameworks *involve models that can plan, reason, and act in iterative loops—*

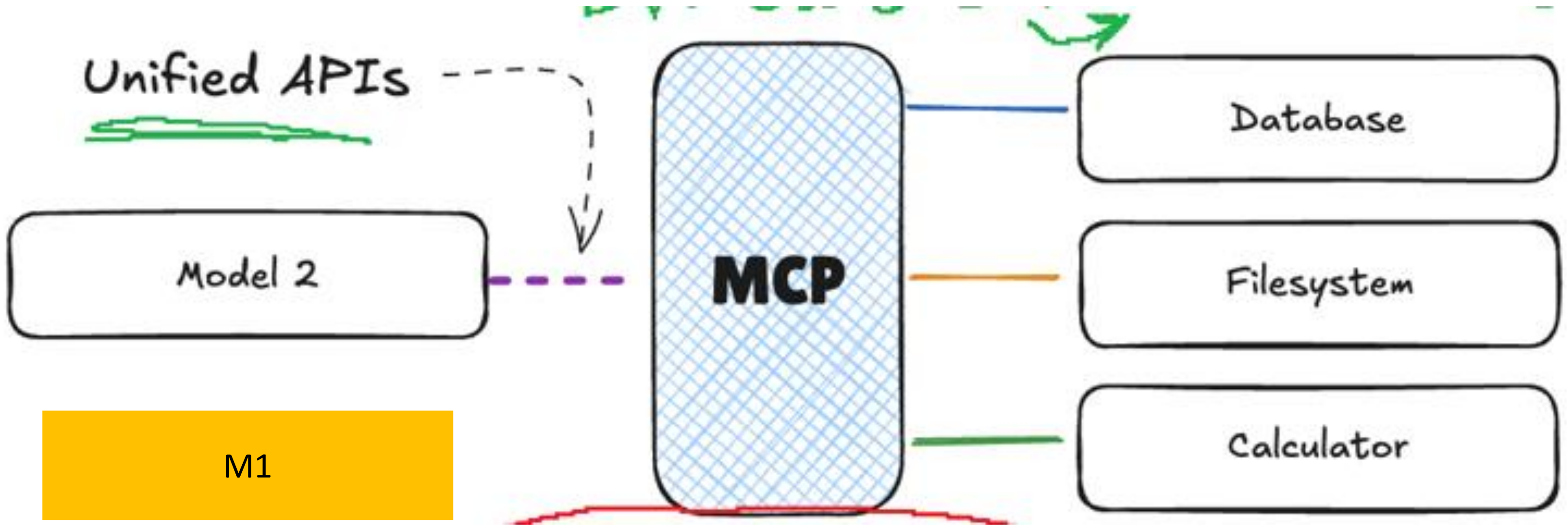
- *they read a task,*
- *think step by step, and*
- *take actions*

The Brain *— LLMs (for reason & Plan)*
Actions *— using Tools*

- *Action – can involve multiple Tools*

Examples:
Hey Siri ? Alexa, Google Assistant
Customer Service - Chatbots
Auto Emails

Where MCP is involved (Recap from MCP Training)



M1

M3

3 x 3 = 9

3 + 3

Agents Demo

AI Agent Observability and Evaluation

1. **Observability** -> understanding what's happening inside your AI agent by looking at external signals
   - like logs, metrics, and traces.

2. For AI agents, this means tracking actions, tool usage, model calls, and responses to debug and improve agent performance.

3. Observability enables:
   - Understand **costs** and accuracy trade-offs
   - Measure **latency**
   - Detect harmful language & prompt injection
   - Monitor user feedback

Some Tools:
https://langfuse.com/

Automated Evaluation Metrics:
https://docs.ragas.io/en/stable/ (For evaluation of LLM application)

AI Agent Observability and Evaluation

2. Evaluation

Common Metrics to Track in Production

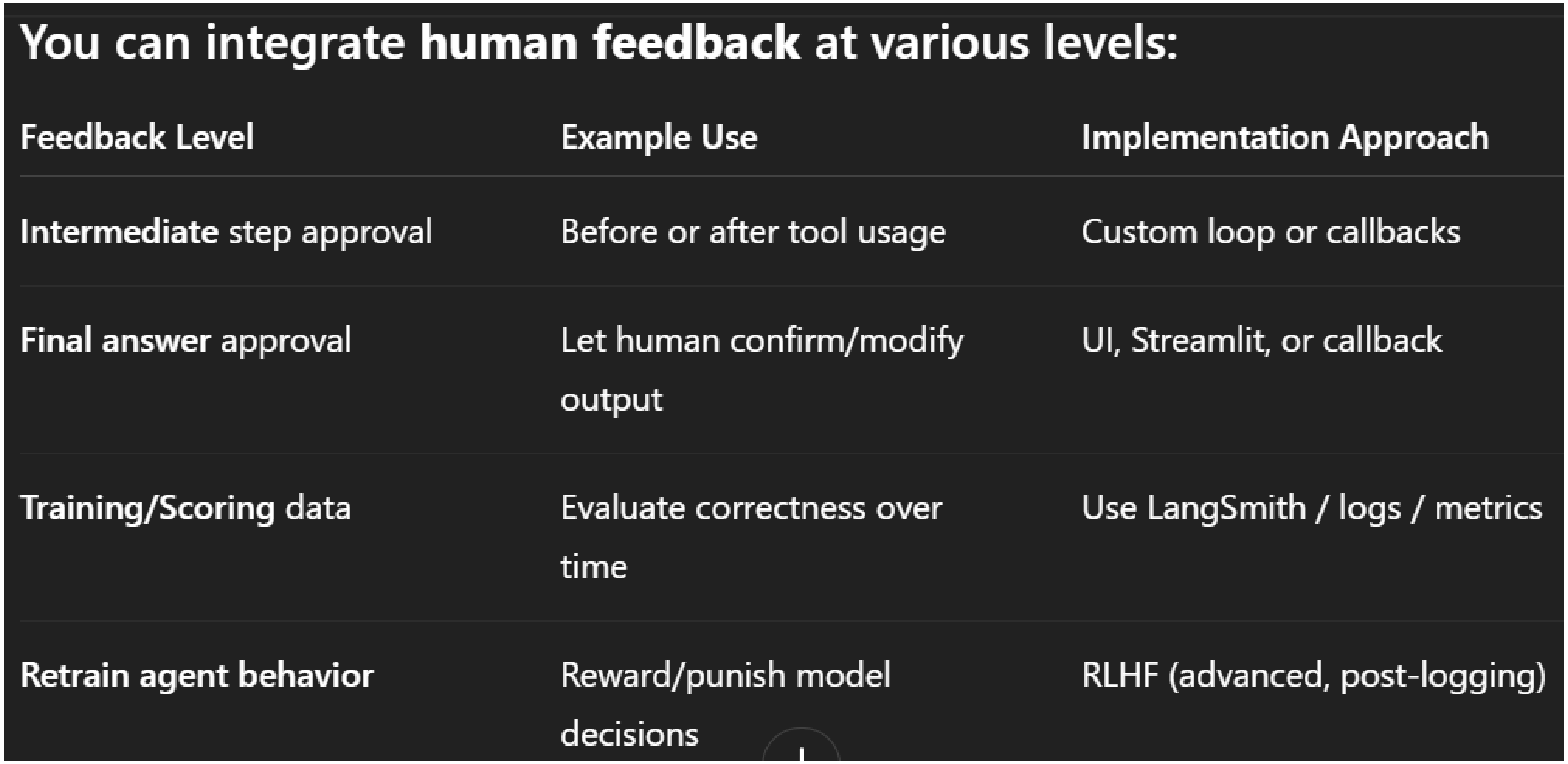
1.Costs — Additional cost for multiple iterations
2.Latency — Observe the time it takes to complete each step, or the entire run.
3.User Feedback — Users can provide direct feedback (thumbs up/down) to help refine or correct the agent.
4.LLM-as-a-Judge — Use a separate LLM to evaluate your agent's output in near real-time (e.g., checking for toxicity or correctness).

Automated Evaluation Metrics:
RAGAS https://docs.ragas.io/en/stable/ (For evaluation of LLM application)

Additional References/Tools from Langchain:

1. Traces with Langchain - https://docs.smith.langchain.com/observability/how_to_guides/trace_with_langchain
2. Langsmith Tutorial on Observability - https://docs.smith.langchain.com/
3. Callbacks can also be used

Human Feedback



You can integrate human feedback at various levels:

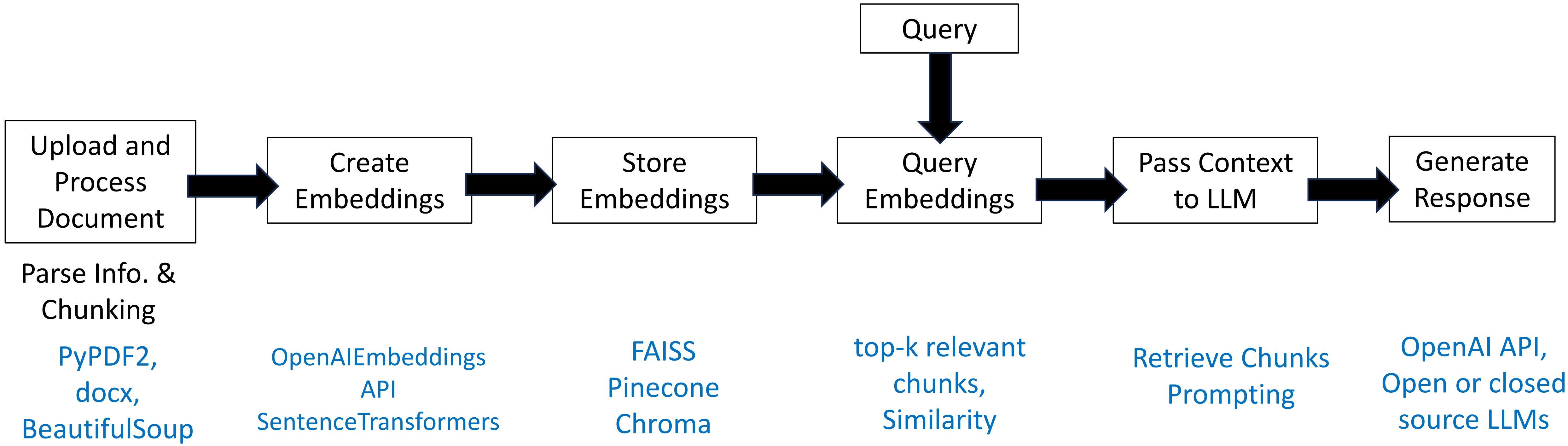
Feedback Level	**Example Use**	**Implementation Approach**
Intermediate step approval	Before or after tool usage	Custom loop or callbacks
Final answer approval	Let human confirm/modify output	UI, Streamlit, or callback
Training/Scoring data	Evaluate correctness over time	Use LangSmith / logs / metrics
Retrain agent behavior	Reward/punish model decisions	RLHF (advanced, post-logging)

An Opensource Library for Agents:

Smolagents: https://huggingface.co/docs/smolagents/en/index

END

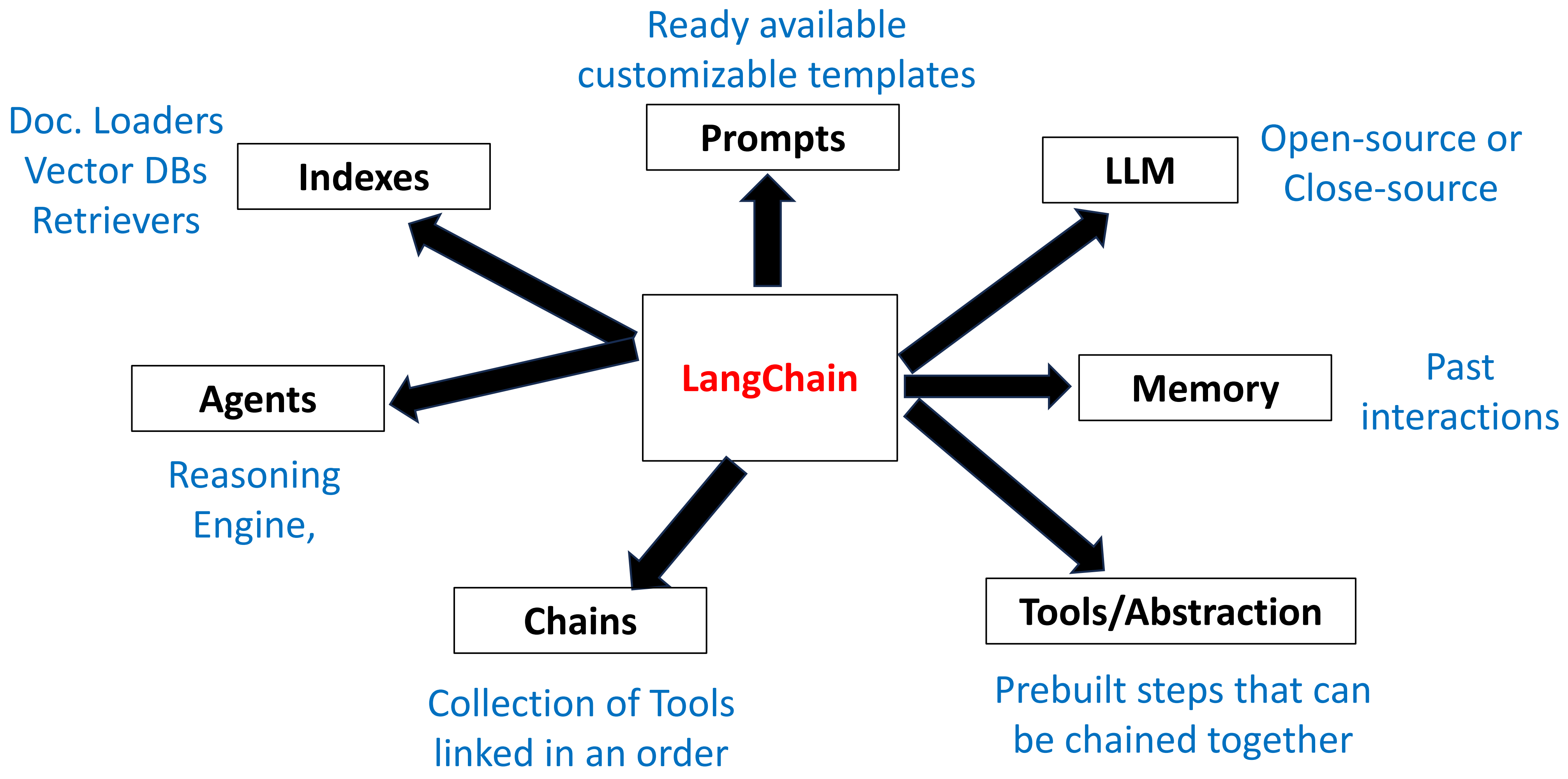
Basic RAG Approach (Recap from Langchain Training)


Query
Upload and Process Document
Create Embeddings
Store Embeddings
Query Embeddings
Pass Context to LLM
Generate Response
Parse Info. & Chunking
PyPDF2, docx, BeautifulSoup
OpenAIEmbeddings API SentenceTransformers
FAISS Pinecone Chroma
top-k relevant chunks, Similarity
Retrieve Chunks Prompting
OpenAI API, Open or closed source LLMs

Langchain can automate the above manual tasks for you

Majorly it can: Interact with External Data, make API Calls, use Memory etc.

LangChain Components


Ready available customizable templates
Prompts
Doc. Loaders
Vector DBs
Retrievers
Indexes
LLM
Open-source or Close-source
LangChain
Memory
Past interactions
Agents
Reasoning Engine,
Chains
Collection of Tools linked in an order
Tools/Abstraction
Prebuilt steps that can be chained together

LangChain Components

1. Tools:

1. **Independent components** that can be chained
   - Embeddings, vector stores, Loaders etc.
2. Interfaces that allow language models to **interact with external systems**, such as:
   - APIs
   - Databases
   - Functions etc.
3. Each Tool has:
   - Name
   - Description
   - Inputs
   - Function
4. **Examples:**
   1. A Function that queries a DB
   2. Call an external API
   3. Initiate a document loader or embeddings

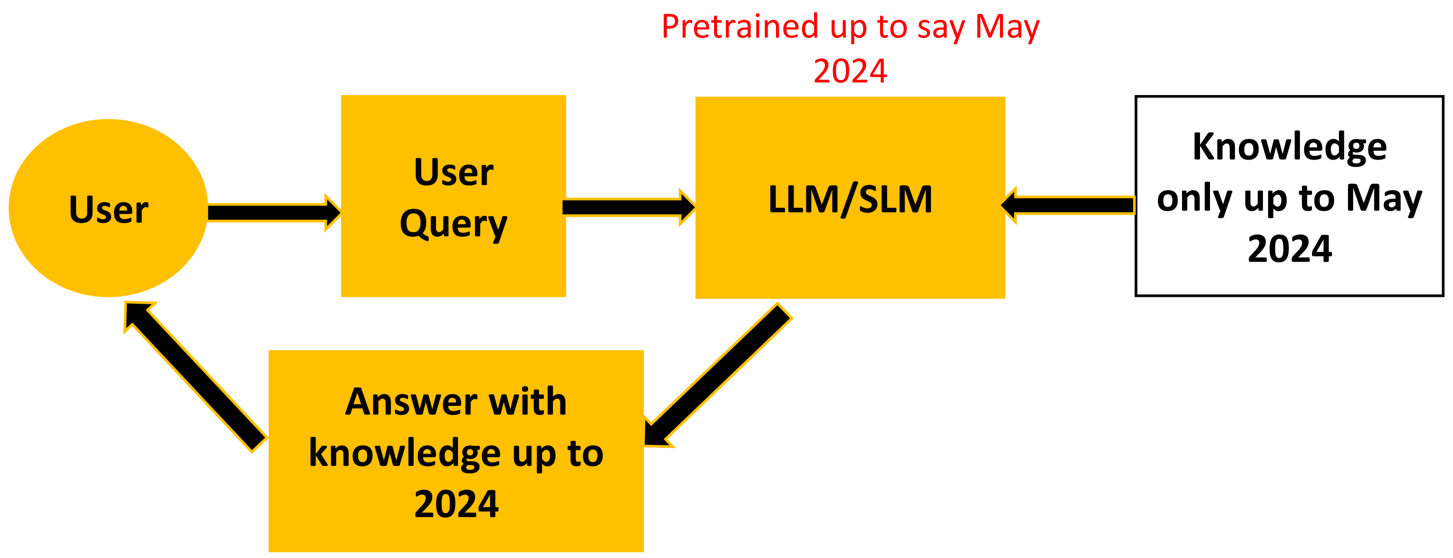
2. Chains:

1. Sequences of Tool calls (or actions): Allows you to **combine multiple tools into a workflow.**
2. Actions in chains are **predefined** and **specified in a specific order**
3. **Linear and Static.** They don't change dynamically based on Input**.**
4. **Examples:**
    1. **Chain1**: <u>A chain for Data/Doc Processing Pipeline</u>: Loading, Embedding, & Storing
    2. **Chain2**: <u>A chain to Query Handling</u>: combine query embeddings + generate response int a chain

3. Agents:

1. <u>Manage workflows</u> – dynamically decide **which Tool/Chain to use** and i**n which order**
2. <u>Agents are responsible for</u>:
    1. Overall logic
    2. Can dynamically adjust the workflow based on input & context
3. **Examples:**
    1. An agent that uses a doc loader, embedding model, vectorstore, & lang model to handle user queries
    2. An agent that process Chain2 after Chain1

Limitation of Pre-trained Models (Recap from MCP Training)


Pretrained up to say May 2024
User
User Query
LLM/SLM
Knowledge only up to May 2024
Answer with knowledge up to 2024

Observations

- Answers are limited to its pretrained ability
- Not open to live or real time information
- For real time info, we need to connect to a tool/agent that can fetch needed information in real time