Multi Tool Langgraph Chain

Why LangGraph?¶

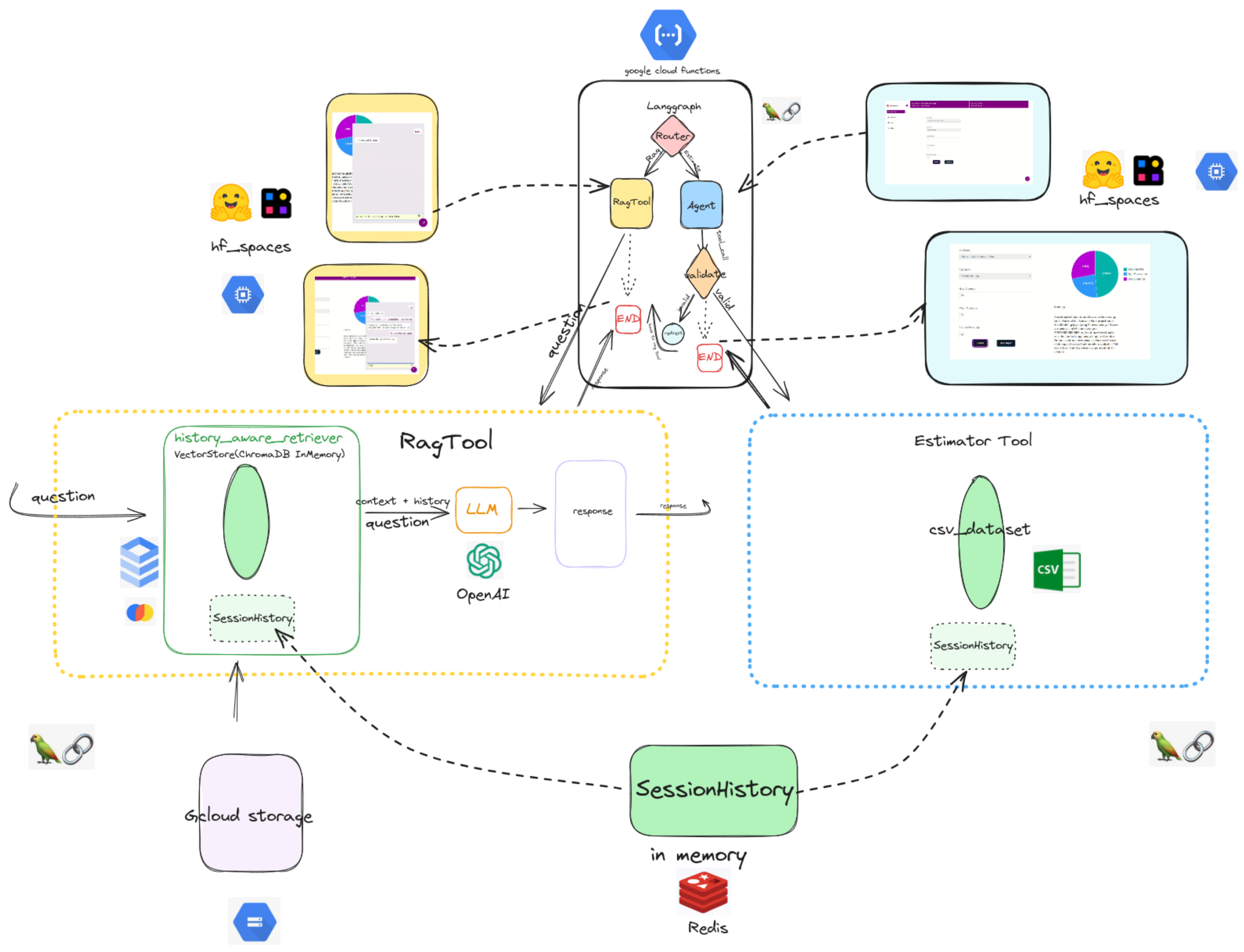
LangGraph is framework agnostic (each node is a regular function). It extends the core Runnable API (shared interface for streaming, async, and batch calls) to make it easy to:

- Seamless state management across multiple turns of conversation or tool usage
- The ability to flexibly route between nodes based on dynamic criteria
- Smooth switching between LLMs and human intervention
- Persistence for long-running, multi-session applications

Key Concepts

- **Stateful Graph:** *LangGraph revolves around the concept of a stateful graph, where each node in the graph represents a step in our computation, and the graph maintains a state that is passed around and updated as the computation progresses.*
- **Nodes:** *Nodes are the building blocks of your LangGraph. Each node represents a function or a computation step. We define nodes to perform specific tasks, such as processing input, making decisions, or interacting with external APIs.*
- **Edges:** *Edges connect the nodes in your graph, defining the flow of computation. LangGraph supports conditional edges, allowing you to dynamically determine the next node to execute based on the current state of the graph.*

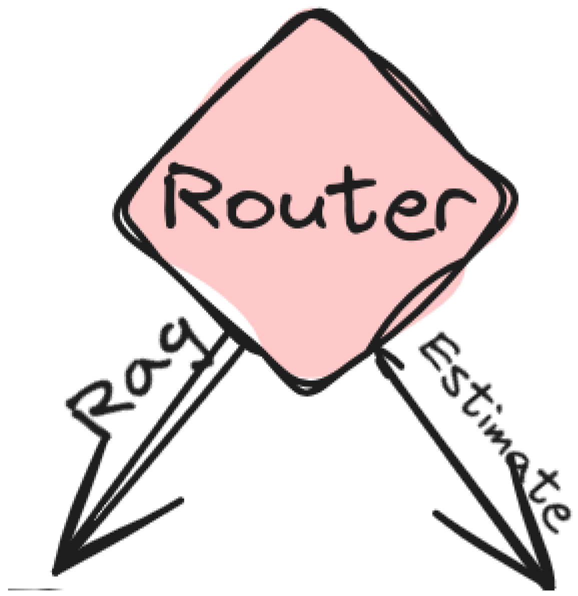
*Lets discuss a use case which uses **langgraph** to utilize multiple tools*



Breakdown of each node of the graph

Router

This node is nothing but a simple LLM router which analyses the query and routes the query to one of the two tools. This node takes in the input query and return either **Rag** or **Estimator** based on which tool is needed to be used



Prompt ━

```
router_system_prompt = """You are an expert at routing a user qu
The estimator_tool is used when the user wants to know the time 
```

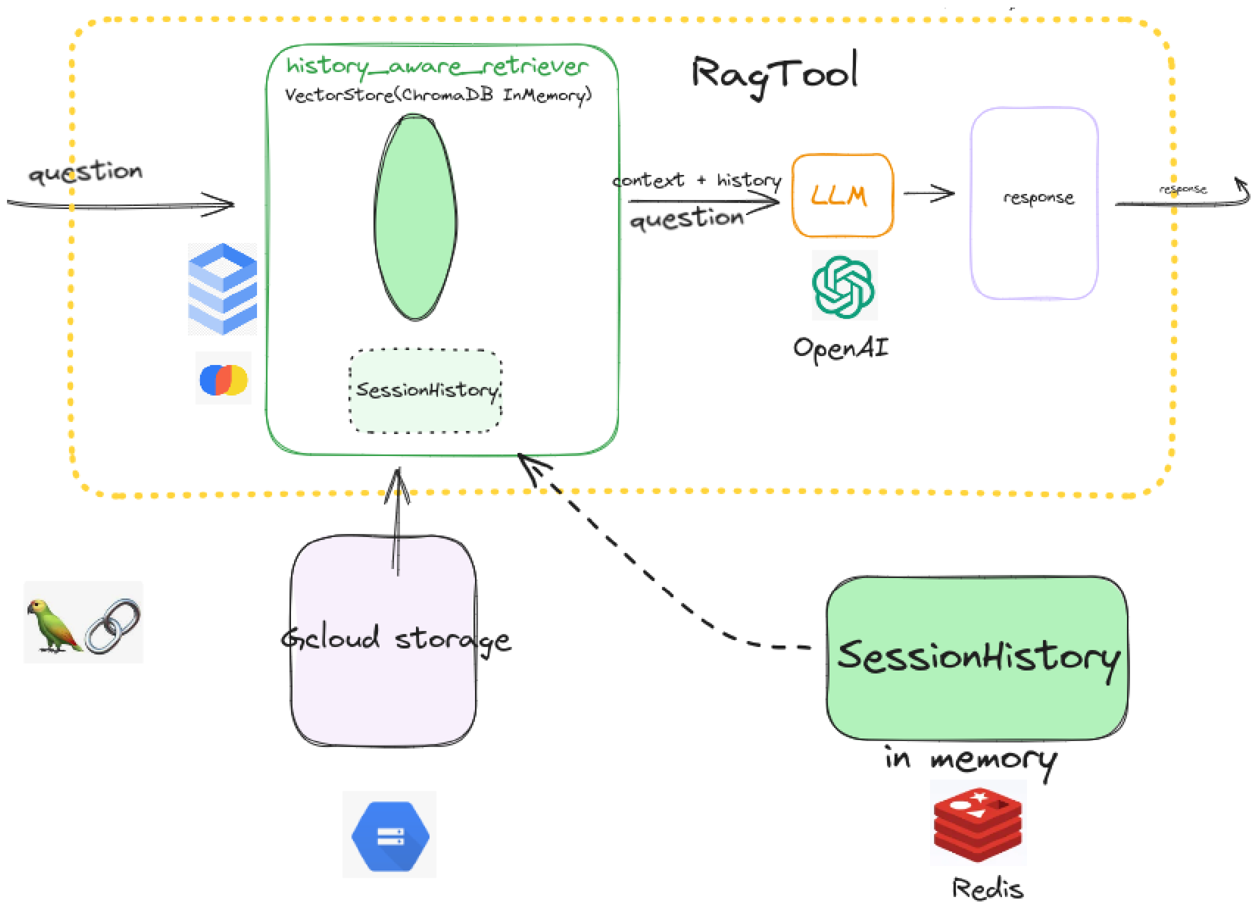
RAGTool -

This node is a langchain tool in itself which is a simple RAG chain which taken in the input , retrieves relevant document from the knowledge base and returns the response

Prompt -

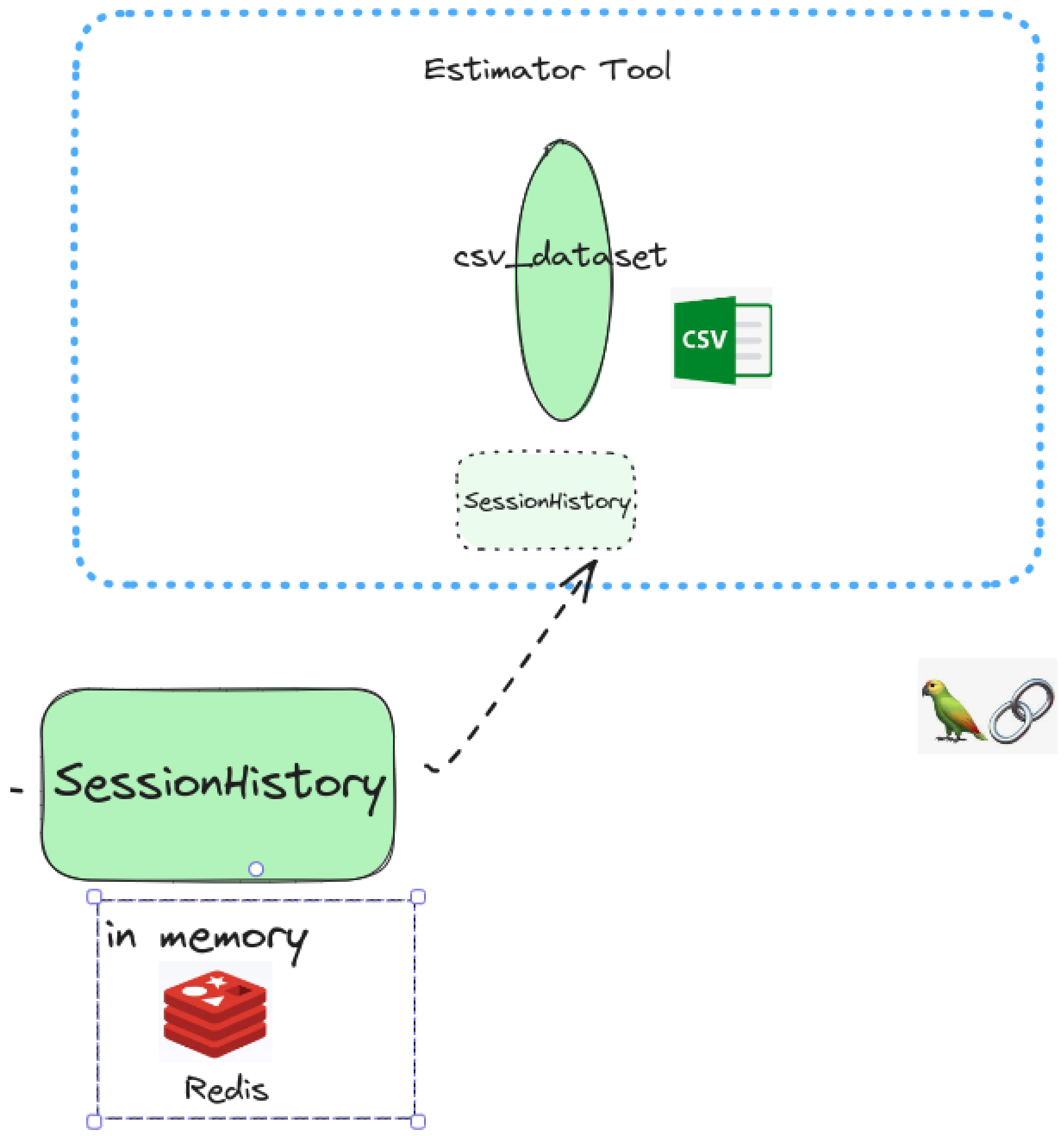
```
qa_system_prompt = """You are an assistant for question-answerin
Use the following pieces of retrieved context to answer the ques
If you don't know the answer, just say that you don't know. \
Use three sentences maximum and keep the answer concise.\

{context}"""
```



EstimatorTool -

This node takes input related to a particular job from the user and gives the amount of time required to complete the modelling of that job



Below is the flow of the graph

- Router is the conditional entry point of the graph
- Based on the response of the router one of the two tools will be invoked.
- If the estimator tool is called then there is an intermediate node in between that is invoked to extract the input parameters from the user query

```
graph = StateGraph(GraphState)

graph.add_node("rag_tool", call_rag_tool)
graph.add_node("agent", call_agent)
graph.add_node("estimator_tool",call_estimator_tool)

graph.set_conditional_entry_point(route_question, {
    "rag": "rag_tool",
    "estimate": "agent"
})

graph.add_conditional_edges("agent",check_estimator_inputs, {
    "valid": "estimator_tool",
    "not_valid": "rag_tool"
})

graph.add_edge("rag_tool", END)
graph.add_edge("estimator_tool", END)
graph.add_edge("generate_answer",END)
```

How LangGraph Improves Upon LangChain

Enables Cyclical Workflows:

- LangChain struggles to represent cycles in its chain definitions, which are crucial for building agent runtimes. These cycles allow the LLM to reason about its next action within the loop, leading to more adaptable and flexible applications.
- LangGraph addresses this limitation by introducing a way to create cyclical graphs, empowering developers to design agent-like behaviors.

Enhances Control and Flexibility:

- While LangChain excels at creating custom chains , it lacks fine-grained control within these chains.
- LangGraph offers more control over how the LLM interacts with various tools, allowing developers to:
- Specify the order in which tools are called.
- Customize prompts based on the application’s state.

Simplifies Complex State Machines:

- Building intricate state machines with LangChain can be cumbersome.
- LangGraph simplifies the process by allowing developers to define state machines as graphs, making them easier to understand and manage.