Agentic AI Application Documentation

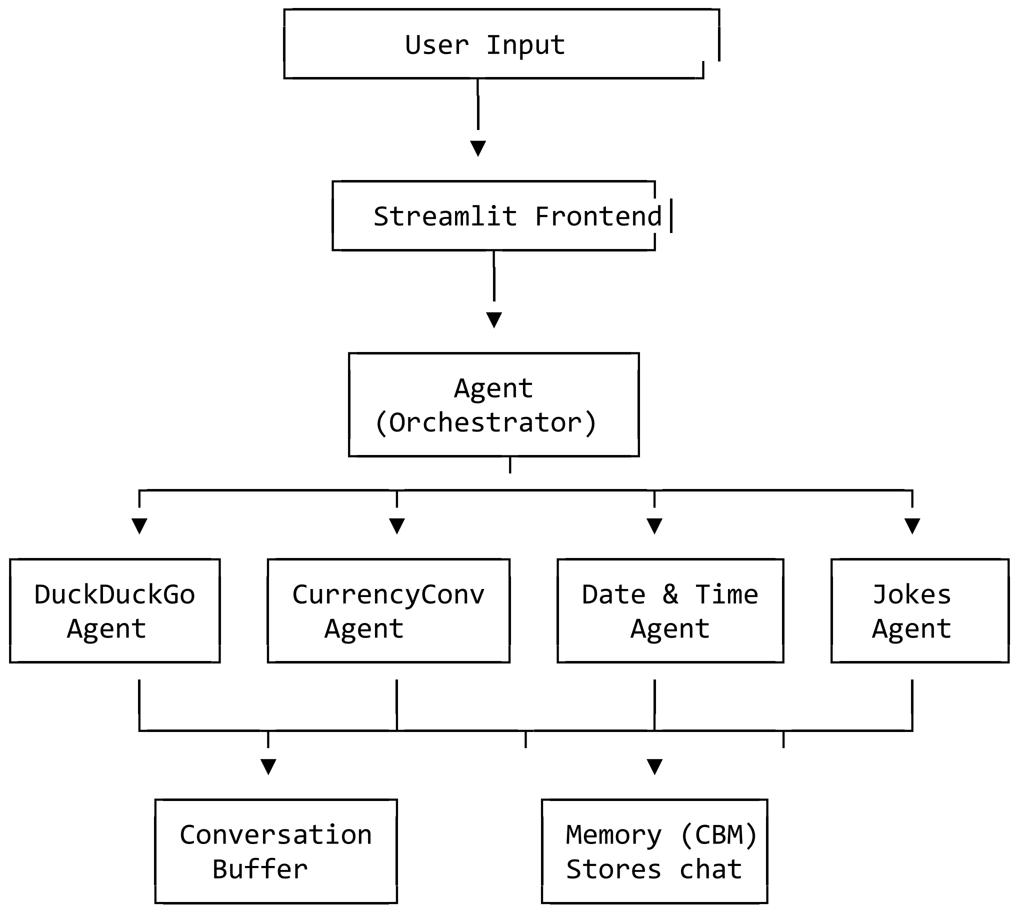
1. Overview

Problem Statement:
This application integrates multiple AI agents to allow users to ask **real-time queries**, and the agents provide **context-aware responses**.
Agents include:
1. DuckDuckGo Search
2. Currency Conversion (USD → INR)
3. Date & Time
4. Jokes

Goal:
To provide users with an interactive, intelligent, and conversational AI experience powered by collaborative agents.

---

2. Architecture



Legend:
- CBM = ConversationBufferMemory
- Memory stores previous messages for context-aware responses.
- Agent acts as the orchestrator: user input -> decides which tool/agent to call -> returns response.

---

3. Agent Collaboration

1. The **Orchestrator Agent** receives user input.
2. It **analyzes intent** using the LLM.
3. Based on intent, it **calls the appropriate tool/agent**:
   - DuckDuckGo → search queries
   - Currency Converter → USD → INR
   - Date & Time → current date/time
   - Jokes → funny responses
4. The Agent updates **ConversationBufferMemory**.
5. Memory allows **context-aware follow-ups**:
   - Example: User asks “Convert 5 USD” → Agent responds
   - Follow-up: “Convert 10 USD now” → Agent remembers previous context

---

4. Instructions to Run

Prerequisites:

- Python
- Streamlit
- LangChain
- Gemini API key using Gemini LLM

Steps:

1. Install dependencies:
```
pip install -r requirements.txt
```

2. Set up **API key** in `.env`:
   ```
   # Gemini API key if using Gemini
   GEMINI_API_KEY=” ”
   ```
3. Run the Streamlit app:
   ```
   streamlit run app.py
   ```
4. Interact with AI agents through the chat interface.

---

5. Notes

- You can **add new agents** by updating `tools.py` and adding them to the `tools` list.
- Memory ensures **long session context** is maintained.
- The architecture supports **scalable multi-agent orchestration**.