RevUp Ai Assessment
Documentation

Problem Statement

Objective: The goal of this assignment is to evaluate the candidate's ability to design and implement a FastAPI-based multi-agent system that utilizes Generative AI (e.g., OpenAI GPT models) for handling multiple tasks in a conversational workflow.

Assignment Details:
Scenario:
You are tasked with building an API that facilitates a multi-agent conversational system to assist users in managing tasks. The system should have the following agents, each specializing in a specific functionality:

1. **TaskPlannerAgent:** Helps users break down tasks into subtasks and prioritize them.
2. **KnowledgeAgent**: Answers factual questions using OpenAI's GPT model.
3. **SentimentAnalysisAgent:** Analyzes user input for sentiment and provides a response accordingly.

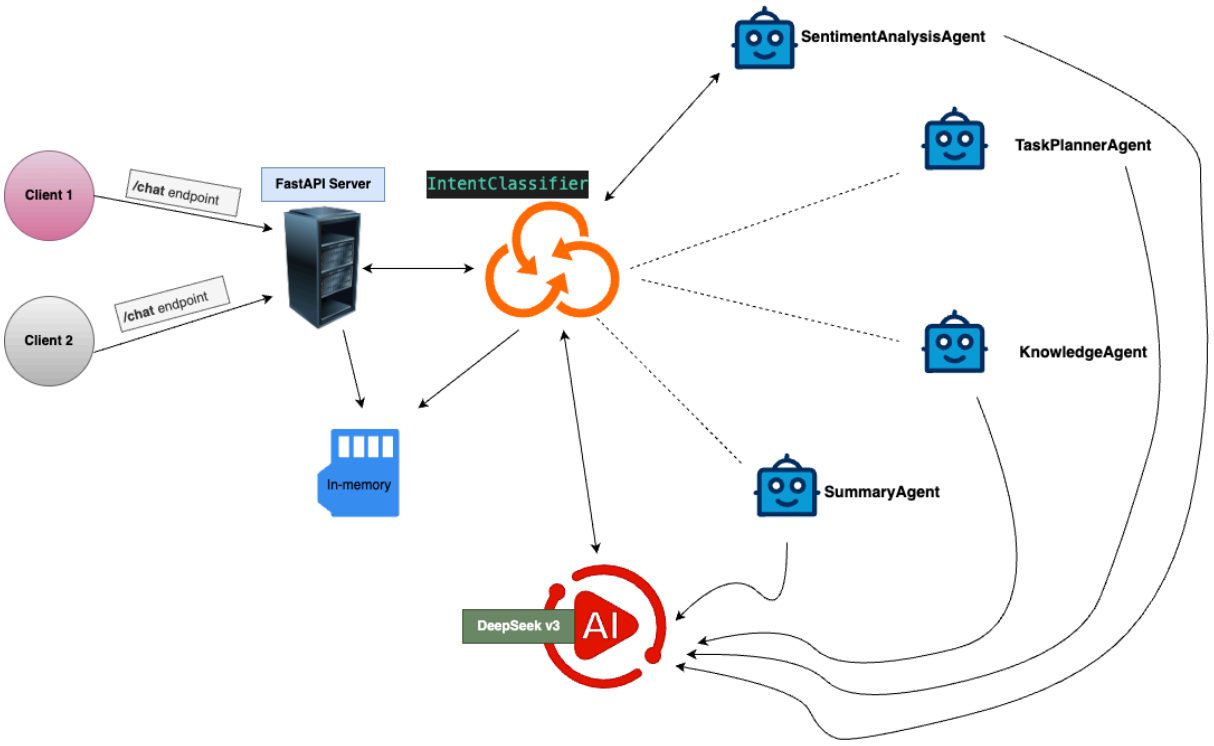
The agents should work together to provide seamless interactions, maintaining context and allowing follow-up queries.

Requirements:

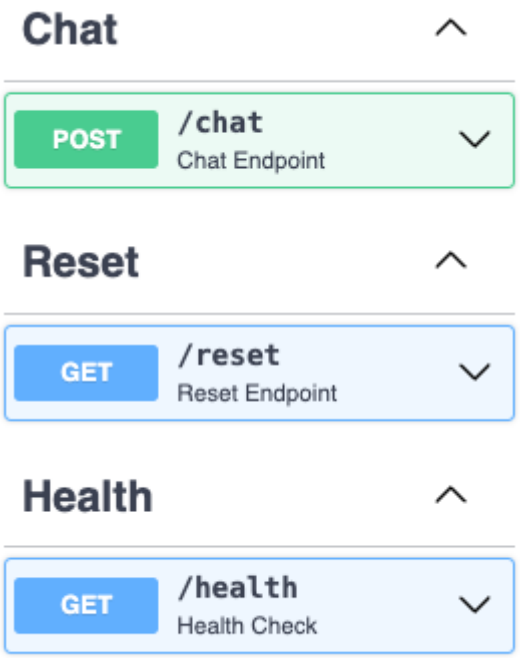
1. **API Implementation:**
    - Use **FastAPI** to create endpoints for user interactions.
    - Implement a root endpoint (`/chat`) that acts as the entry point for user queries.
2. **Agent Workflow:**
    - Design the application to route user input to the appropriate agent(s) based on the query type.
    - Implement a context management mechanism to maintain user session and conversation state.
3. **Integration with OpenAI:**
    - Use OpenAI's GPT model to generate responses for:
        - Task breakdowns (TaskPlannerAgent).
        - Answering general knowledge questions (KnowledgeAgent).
        - Sentiment analysis feedback (SentimentAnalysisAgent).
4. **Key Features:**
    - **Intent Classification:** Implement an intent classifier to determine which agent should handle the user's query.
    - **Context Awareness:** Ensure agents can handle follow-up questions by maintaining conversation history.
    - **Error Handling:** Include error responses for invalid queries or API failures.

5. **Expected Endpoints:**
    - `POST /chat`: Accepts a user query and routes it to the appropriate agent(s).
    - `GET /status`: Returns the status of the application (health check).
    - `POST /reset`: Resets the conversation context for a user session.
6. **Optional (Bonus):**
    - Extend the system to add a **SummaryAgent** that summarizes the conversation at the end of a session.
    - Implement a logging mechanism to track agent interactions and responses.

Architecture Diagram



API Endpoints



Intent Classification and Context Management

- **Develop an IntentClassifier class** - an Orchestrator that uses the Fireworks API to classify the intent of the user query, and decides the specific agent(s) that should handle the query based on the intent. The class is responsible for invoking the appropriate agent(s) to handle the query. It provides a JSON response like

```
{'agent_name_1':'associated query', 'agent_name_2':'associated query'}
```

- **Develop a SessionManager class** that manages the context of the conversation with the user. It maintains the state of the conversation, and the context of the conversation with the user by storing the conversation history in-memory. Ideally, it should be a persistent storage like MongoDB.

Agents

1. TaskPlannerAgent: Helps users break down tasks into subtasks and prioritize them.
2. KnowledgeAgent: Answers factual questions using OpenAI's GPT model.
3. SentimentAnalysisAgent: Analyzes user input for sentiment and provides a response accordingly.
4. SummarizationAgent: Summarizes the conversation with the user.

Libraries and Frameworks

- **FastAPI:** A modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints.
- **Fireworks API:** A platform that provides Open-source large language models deployed as RESTful APIs.
- **DeepSeekV3:** DeepSeek 128K Context model provided by DeepSeek AI.
- **LangChain:** A library that provides a simple interface to initialize agents and interact with them.
- **Pydantic**: Data validation and settings management using Python type annotations.
- **Uvicorn:** A lightning-fast ASGI server implementation, using uvloop and httptools.