Sales Copilot

Objective:

The goal of the Sales Copilot project is to create an intelligent system that processes user input in the form of text queries and provides relevant, actionable responses based on the available data. The system leverages tools for structured and unstructured data retrieval to generate dynamic and contextually accurate Responses.

Work Done Till Now:

1. **Agent Implementation with LangChain:**
    - I implemented an agent using LangChain, which serves as the core decision-maker for selecting the appropriate tool based on user input.
    - The agent processes the input text and decides whether to use the **SQL Tool** or the **Unstructured Data Tool** to retrieve the most relevant information.
2. **SQL Tool Implementation:**
    - The **SQL Tool** is responsible for querying structured data stored in a PostgreSQL database.
    - **Prompt Engineering:**
      I defined a prompt that includes a clear schema of the tables and columns in the database. This prompt serves as the context for the tool to understand the structure of the data.
    - **Keyword Matching:**
      The tool identifies relevant keywords from the user input and uses them to search for the corresponding tables in the database schema.
    - **SQL Query Generation:**
      The agent leverages the **Gemini model** to generate an appropriate SQL query based on the identified tables and user query.
    - **Data Retrieval:**
      Once the SQL query is generated, the PostgreSQL database is queried, and relevant data is fetched to provide the response.
3. **Unstructured Data Tool Implementation (RAG):**

- For unstructured data, I implemented **Retrieval-Augmented Generation (RAG)**, which uses a combination of a retrieval mechanism and language generation for providing answers.
- **Data Retrieval:**
  Based on the user query, relevant documents or chunks of information are retrieved from the data source.
- **Response Generation:**
  The retrieved data is used to generate an appropriate response that addresses the user's query.
- **Tabular Data in Vector Database:**
  I stored tabular data in a **vector database** to enhance the system's capability to answer reasoning-based queries. When such a query is identified, the **Unstructured Data Tool** retrieves relevant information from the vector database and generates a suitable response.

4. **Handling Larger Databases:**
   - I successfully handled a database containing **3 tables with 20,000 to 30,000 rows** of data.
   - The system is capable of retrieving responses in **under 10 seconds** for most queries, ensuring efficiency.
   - **Performance:**
     However, after processing **2-3 consecutive inputs**, there is a slight slowdown in the response time. Further optimizations are being explored to address this minor performance issue.
5. **Final Response Generation:**
   - After the relevant data is fetched from either the SQL Tool or the Unstructured Data Tool, the agent generates a final response that incorporates the findings from the chosen tool.
   - The agent ensures that the response is contextually accurate and helpful based on the input query.

Challenges and Current Work:

- **Issue:**

  Some user queries are not returning the expected answers, despite using both the SQL and Unstructured Data Tools effectively.
- **Work in Progress:**

  I am currently focusing on improving the system's ability to choose the right tool based on a broader range of queries. This includes refining the agent’s decision-making process and enhancing the retrieval logic to provide more relevant data for complex queries.
- **Future Steps:**
  - Fine-tune the tool selection process by improving keyword extraction and relevance matching.
  - Enhance the retrieval mechanism for unstructured data, ensuring more accurate document matching.
  - Optimize the overall agent workflow for faster and more efficient responses, especially for consecutive queries.

Technologies Used:

- **LangChain:** For building the agent that selects the appropriate tool.
- **PostgreSQL:** For structured data storage and retrieval.
- **Gemini:** For generating SQL queries based on a defined prompt.
- **Pinecone:** For storing tabular data to handle reasoning-based queries effectively.
- **RAG:** For unstructured data retrieval and response generation.

Below are the Output examples

Dynamic Document and Query Manager

Ask a question:

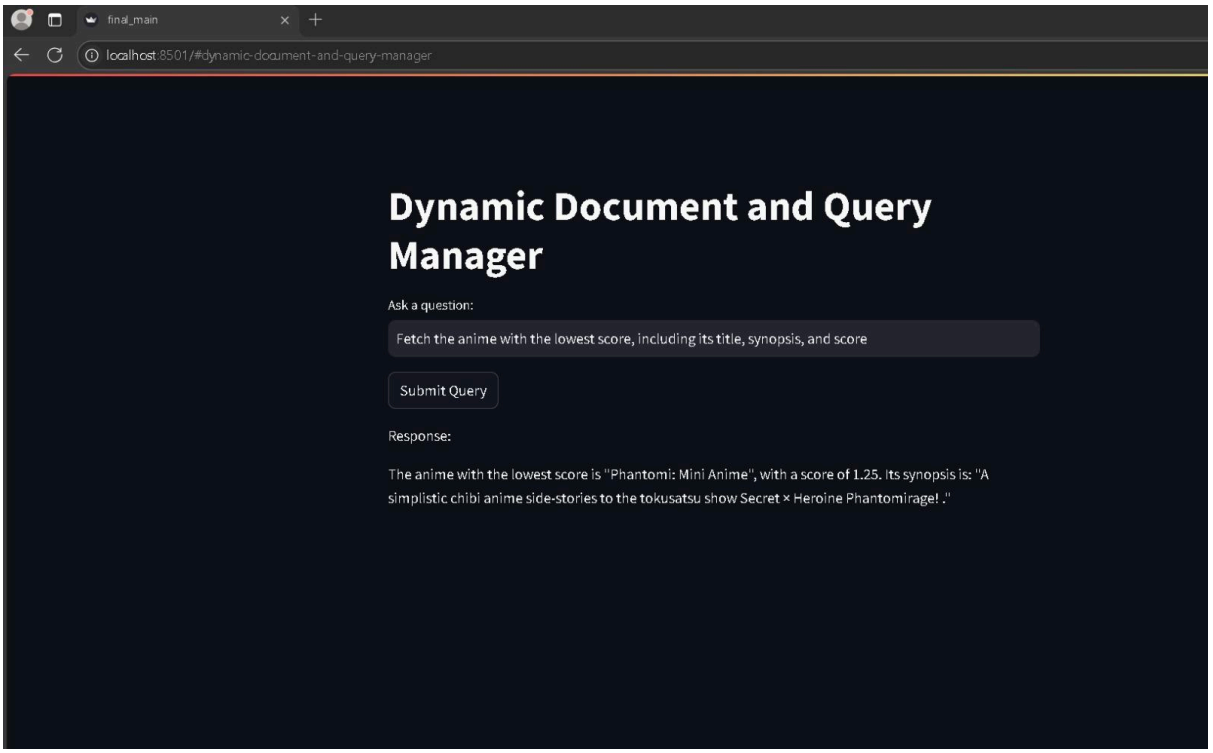
Which are the top 5 animes with the highest number of members associated with them?

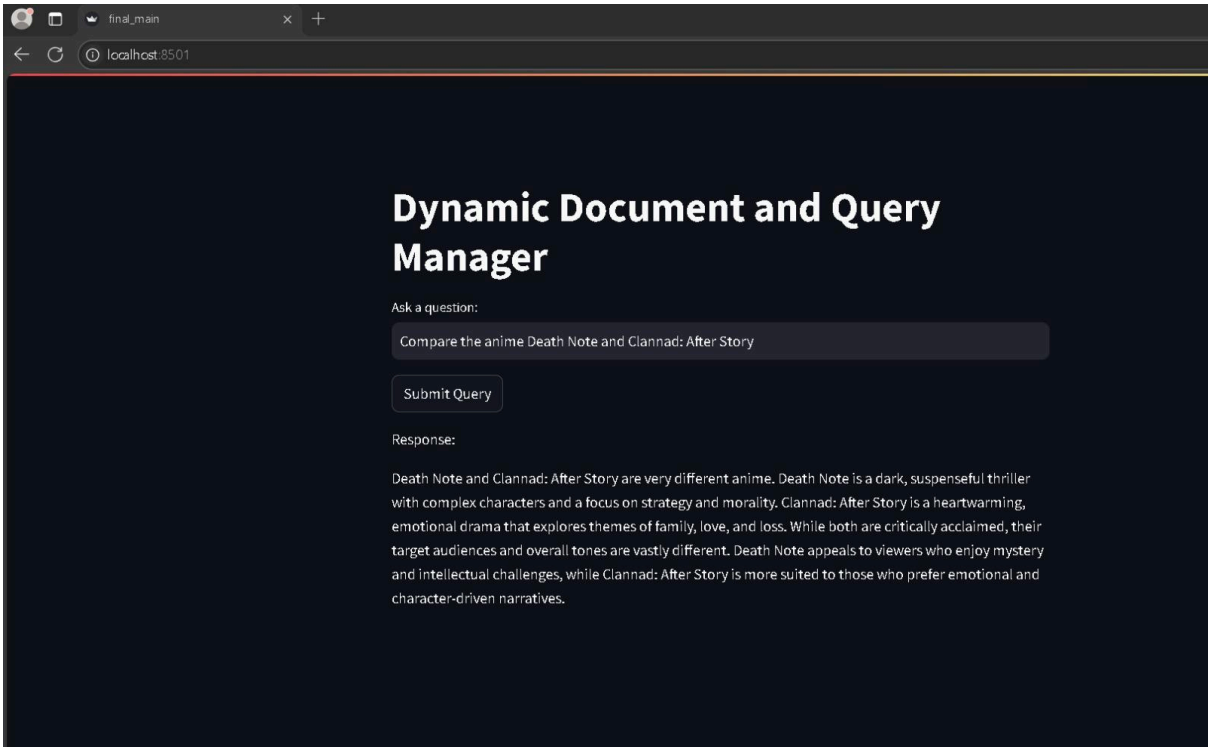
Submit Query

Response:

The top 5 animes with the highest number of members are:

1. Death Note (3416 members)
2. Steins;Gate (3116 members)
3. Kimi no Na wa. (2872 members)
4. Fullmetal Alchemist: Brotherhood (2548 members)
5. Clannad: After Story (2388 members)




final_main
localhost:8501
Dynamic Document and Query Manager
Ask a question:
Compare the anime Death Note and Clannad: After Story
Submit Query
Response:
Death Note and Clannad: After Story are very different anime. Death Note is a dark, suspenseful thriller with complex characters and a focus on strategy and morality. Clannad: After Story is a heartwarming, emotional drama that explores themes of family, love, and loss. While both are critically acclaimed, their target audiences and overall tones are vastly different. Death Note appeals to viewers who enjoy mystery and intellectual challenges, while Clannad: After Story is more suited to those who prefer emotional and character-driven narratives.