Movie Chatbot Development Process

1. Domain Selection

Application: Movie Chatbot

Scope and Data Type:

- The application will handle data related to movies, including movie overview, genres, release dates, and ratings.
- The chatbot will respond to user queries about movies, provide recommendations, and offer information about movie details.

2. Data Collection and Preprocessing

Data Source: Kaggle dataset on movies, Prompt Engineering to get mock data

Steps:

1. **Acquire Data:** Download the movie dataset from Kaggle.
2. **Data Cleaning:**
    - Remove any irrelevant columns or information not needed for the chatbot.
    - Handle missing data by filling in defaults or removing incomplete entries.
    - Generate missing plots of movies using LLM.
3. **Text Preparation:**
    - Remove special characters and unnecessary whitespace.
    - Address missing by prompt - engineering.

3. Vector Database Implementation

Chosen Database: Pinecone

Steps:

1. **Setup Pinecone:**
    - Create an account and set up an index in Pinecone.
2. **Data Indexing:**
    - Use sentence transformer to Transform movie descriptions and relevant textual data into embedded vectors.
    - Store these vectors in Pinecone, ensuring efficient indexing for quick retrieval.
3. **Configuration:**
    - Configure the index to support semantic similarity searches, which will allow the chatbot to retrieve relevant movie data based on user queries.

4. Application Development

Frameworks Used: FastAPI for backend, Streamlit for frontend

Steps:

1. **User Interface:**
    - Designed a simple, user-friendly interface in Streamlit where users can input natural language queries.
2. **Backend Logic:**
    - Implemented a FastAPI backend to handle incoming queries
    - Based on chat history, refine latest user question to standalone question based on chat history. Use a GPT-3.5 instruct to understand user intent and extract relevant information.
    - Use the Pinecone vector database to retrieve the most relevant movie data based on the processed query
    - Embed the retrieved context and refined query into prompt in lanchain and send the content to user.
3. Orchestration: orchestrated frontend and backend using docker-compose.

5. Evaluation and Testing

Steps:

1. **Testing:**
    - Conduct extensive testing with a variety of queries to ensure the chatbot understands and responds accurately.
    - Include edge cases and unusual queries to evaluate robustness.
2. **Performance Evaluation:**
    - Measure response time and accuracy of the chatbot.
    - Fine-tune the model and indexing parameters to optimize performance.

Challenges Faced

Data Quality:

Challenge: The Kaggle dataset contained incomplete or inconsistent entries.

Solution: Implemented robust data cleaning procedures, including filling missing values and removing irrelevant information.

Vector Database Configuration:

Challenge: Ensuring efficient and accurate indexing in Pinecone.

Solution: Experimented with different indexing parameters and configurations to optimize performance for semantic similarity searches.

Query Understanding:

Challenge: Ensuring the LLM accurately understands and processes diverse natural language queries.

Solution: Tuned prompt of LLM using a variety of techniques learnt in class to improve understanding and accuracy.

Performance Optimization:

Challenge: Maintaining fast response times while handling complex queries.

Solution: Tested latency against different retreival mechanisms cosine, dot product and euclidian for latency