Agent Systems Project: Conversational Agent Using LangChain

Project Overview

This project involves building a Conversational Agent that utilizes Large Language Models (LLMs) with frameworks such as LangChain. The agent will be capable of engaging in natural conversations with users, responding to their queries based on context-aware reasoning.

Technology Stack:

- Programming Language: Python
- LLM Frameworks: LangChain
- LLM Model: ChatOpenAI (gpt-3.5-turbo), OpenAIEmbeddings (text-embedding-ada-002)
- Database: Vector Databases (FAISS)
- Deployment: Flask
- UI: Streamlit

System Architecture:

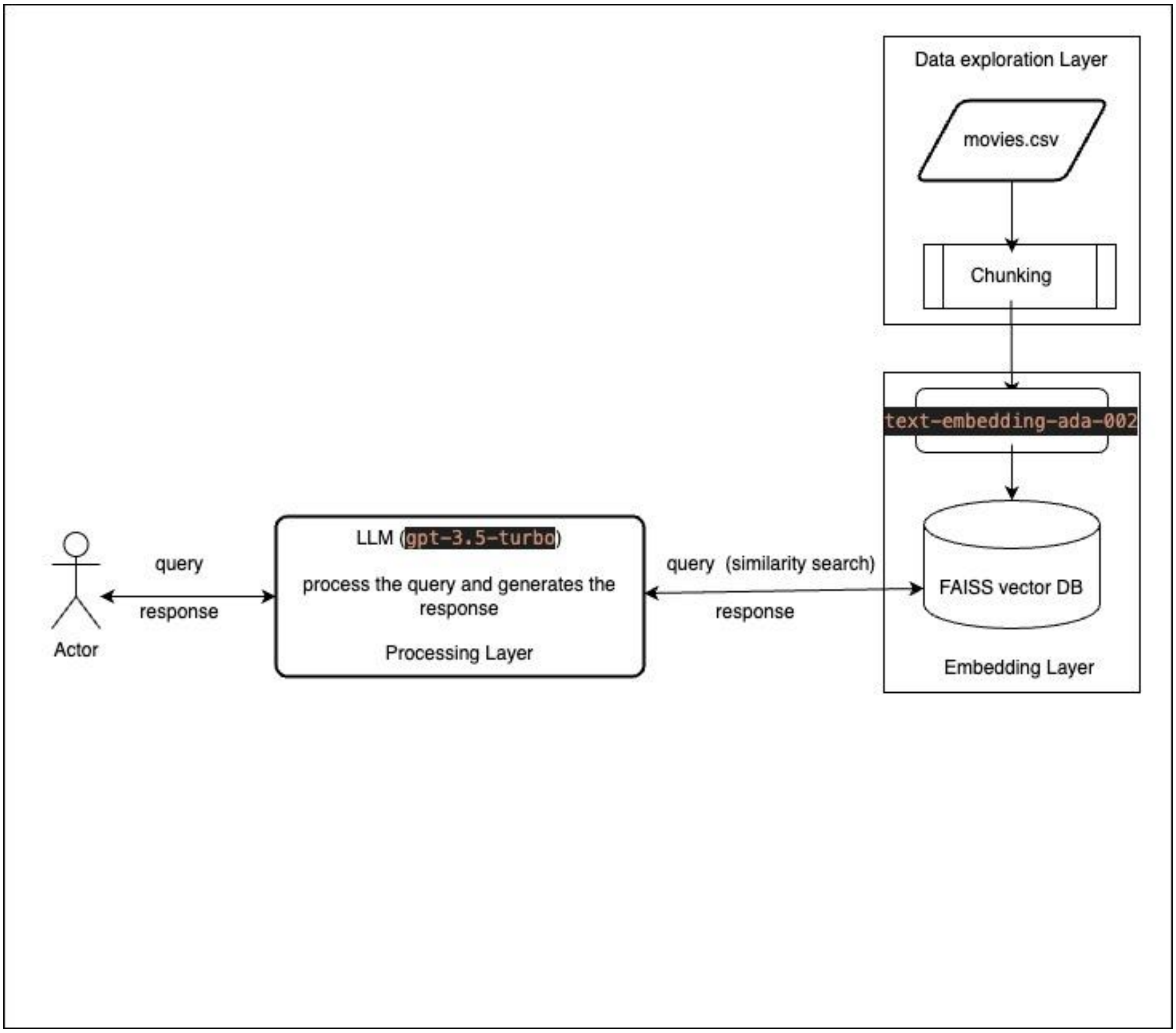
User Query → Input text is processed.

LLM Processing → LangChain processes input using context & memory.

Data Retrieval → FAISS fetches relevant information.

Response Generation → LLM generates meaningful responses.

User Interaction → Chatbot provides responses & learns from interactions.



Project Functionalities:

1. User Query Processing
   - Accepts user input via text
   - Tokenizes and processes queries
   - Detects user intent for context-aware responses

2. Context Management
    - Maintains conversation history
    - Stores previous queries for contextual reasoning
    - Uses LangChain memory components

3. Information Retrieval using FAISS
    - Fetches relevant documents from external sources
    - Indexes knowledge bases for rapid search
    - Enhances response accuracy with up-to-date information

4. Response Generation with LLMs
    - Uses LangChain prompt engineering
    - Generates responses based on context and knowledge
    - Dynamically adapts to conversation flow

Integration with External APIs:

- Supports web search, databases, and document retrieval
- Fetches real-time data from APIs (e.g., news, weather)
- Uses OpenAI and third-party plugins for enhanced responses

Agent Capabilities & Reasoning:

- Uses LangChain Agents for advanced decision-making
- Implements tools such as search, calculations, and summarization
- Adapts to multi-turn conversations with logical reasoning

UI & Deployment:

- Provides a user-friendly chat interface (Streamlit)
- Deploys via Flask for backend processing