Final Project Milestone 02: Architecture and Implementation

AI Interview Coach – PrepBot

Sridhar Cheppala
Brunda Katragadda
Tirumala Arikatla

April 2025

Team Name

PrepTech

Project Topic

AI Interview Coach – PrepBot

Project Tools and Technologies

- **Python 3.11** – Core programming language for backend development.
- **LangChain** – Manages LLM-based chaining, entity routing, and document retrieval.
- **OpenAI GPT-3.5 / GPT-4** – Generates answers and performs NER from user questions.
- **Neo4j** – Graph database for storing structured interview topics and relationships.
- **Streamlit** – Web frontend for interactive chatbot interface.
- **Google Colab** – Cloud-based development environment.
- **LocalTunnel** – Exposes Streamlit app publicly for testing/demo.
- **Tiktoken / OpenAI Embeddings** – Embeds and indexes text documents for similarity search.

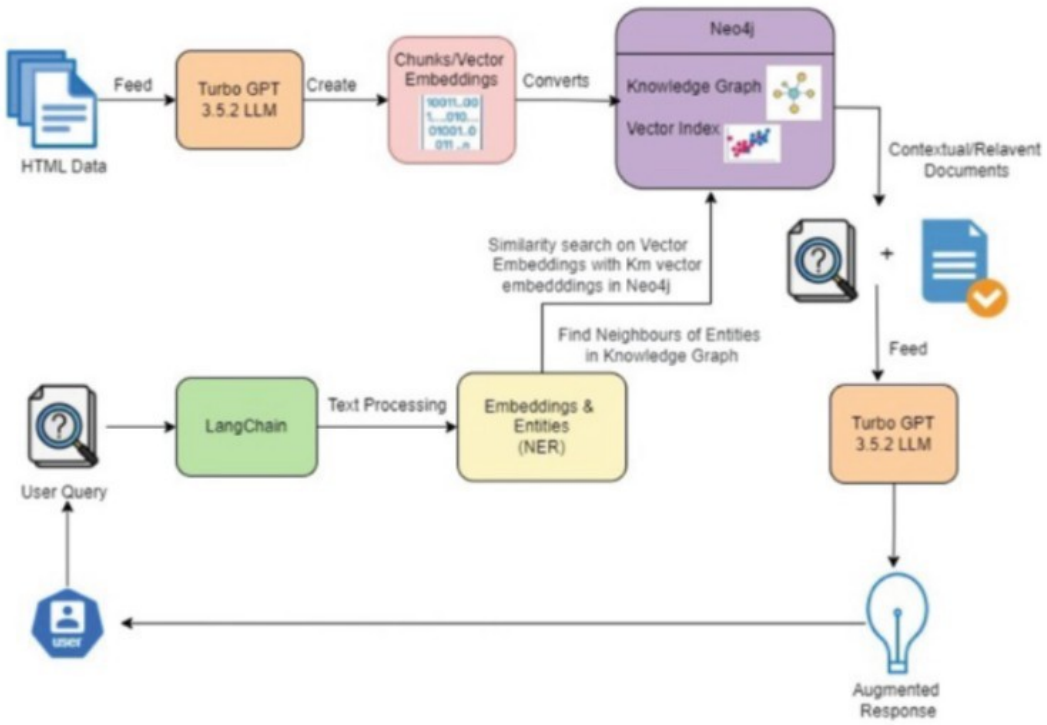

Figure 1: System Architecture of PrepBot – AI Interview Coach

High-Level Architecture Diagram

Architecture Explanation

- **HTML Data Feed**: Interview-related content is loaded and preprocessed.
- **Embedding Engine**: Input content is split and embedded using OpenAI embeddings.
- **Neo4j Graph**: Stores structured entities like questions, categories, and relations.
- **LangChain Engine + NER**: Extracts relevant terms using LLM and performs graph/vector search.
- **Retriever**: Retrieves context from both structured (graph) and unstructured (vector) sources.
- **LLM Response Generator**: Uses GPT-3.5 to generate accurate, natural interview answers.
- **Streamlit Frontend**: Provides a chatbot interface to end-users.

Implementation Plan

1. **Environment Setup**

- Install all required libraries (LangChain, Neo4j, Streamlit, OpenAI, etc.)
- Configure environment variables for API keys and Neo4j connection.

2. **Frontend UI with Streamlit**
   - Build a simple chat interface to enter questions and display answers.
   - Connect it to the backend for real-time interactions.
3. **Document Collection and Preprocessing**
   - Scrape and clean interview questions using WebBaseLoader.
   - Chunk documents and embed using OpenAI embeddings.
4. **Graph Construction in Neo4j**
   - Use LangChain's LLMGraphTransformer to extract entities and build relationships.
   - Store structured entities like topics, question types, and answers in Neo4j.
5. **Retriever + Chain Logic**
   - Combine Neo4j full-text search and embedding vector search to get context.
   - Feed into prompt chain with GPT to generate the final answer.
6. **Answer Generation**
   - Use LangChain to format context and questions and invoke GPT-3.5.
   - Format output and return it to the frontend.
7. **Testing and Deployment**
   - Test with 30+ mock questions.
   - Use LocalTunnel to expose the Streamlit app.