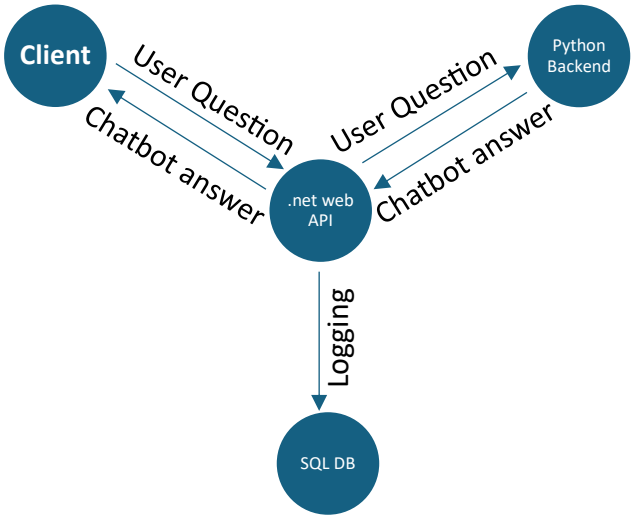
LDC Technical Assessment

Imagine you're building the backbone of an intelligent chatbot system that seamlessly integrates two worlds: **.NET Web API** and **Python AI backend**. The goal is to create a **RAG (Retrieval-Augmented Generation)** chatbot that processes user questions and retrieves accurate answers from provided text documents.

The chatbot will operate like this:

1. A user asks a question.
2. The .NET API forwards the question to the Python backend.
3. The Python backend processes the query using LangChain, retrieves relevant information from the documents, generates a response, and sends it back.
4. The .NET API returns the response to the user while logging all interactions in an SQL database for tracking and analysis.



This task combines cutting-edge AI frameworks, robust backend engineering, and database management to create a user-friendly chatbot.

Requirements

1. .NET Web API

- Build a **.NET Web API** that:
  - Accepts user questions via HTTP POST **asynchronous** requests.
  - Forwards these questions to the Python backend over HTTP.
  - Receives the chatbot's responses from the Python backend.
  - Logs user queries and chatbot responses into an **SQL database**.

2. Python Backend

- Develop a Python backend using the **LangChain Framework** to:
  - Embed and index the provided text document for retrieval.
  - Implement a **RAG pipeline**:
    - **Retrieve** relevant chunks from the documents based on the user's question.

        - **Generate** a response using a language model (e.g., OpenAI's GPT API, Ollama Llama, etc.).
    - Return the generated response to the .NET API.

3. SQL Database

- Design an **SQL database** to store user-chatbot interactions:
    - UserQueries: user questions with timestamps.
    - ChatbotResponses: chatbot responses with timestamps.
- Ensure that each query-response cycle updates the database.

Deliverables

1. **.NET Web API Code**:
    - A fully functional API with endpoints to accept **asynchronous POST requests** containing user queries.
    - Includes logic for forwarding requests, receiving responses, and database logging.
2. **Python Backend Code**:
    - A LangChain-based implementation for document retrieval and response generation.
    - Uses text embeddings and an LLM to ensure accurate and meaningful answers.
3. **SQL Database Schema**.

Note:
You can find 2 attached txt document with dataset you will use to build a chatbot based on it.

Conclusion

This task showcases your ability to integrate **AI systems with .NET APIs**, use **LangChain for RAG pipeline**, and manage data with an SQL database. Feel free to ask questions or request clarification—our team is here to assist you as needed in our discord channels.

Good luck, and we look forward to your amazing contribution!