AI Engineer (Level-1) — Technical Assessment

Problem Statement:
Develop a Simple Multilingual Retrieval-Augmented Generation (RAG) System.

Source Code:
Github repo Link: Multilingual_Rag
Git clone: git@github.com:sadia4444a/Multilingual_Rag.git

Setup guide:

🚀 Step-by-Step Setup

1. Clone the Repository

```
git clone
git@github.com:sadia4444a/Multilingual_Rag.git
cd Multilingual_Rag
```

2. Install Poetry (if not installed)

```
# For Linux/macOS
curl -sSL https://install.python-poetry.org | python3
-
```

```
brew install poetry

# Or for Windows (PowerShell)
(Invoke-WebRequest -Uri
https://install.python-poetry.org
-UseBasicParsing).Content | python -
```

3. Install project dependencies:

```
poetry install
```

4. Setup Environment Variables:

```
OPENAI_API_KEY=your-api-key
```

5. Run the Project

```
poetry run streamlit run app.py
```

Or

```
poetry shell
streamlit run app.py
```

Used tools, library, package:

Python (programming language)
LangChain (for language model pipelines, retrieval, embeddings, and text splitting)
OpenAI API (via `langchain_openai` and `OpenAIEmbeddings`)
FAISS (for efficient vector similarity search)
PDF tools (`multilingual-pdf2text`, `unstructured`) for PDF text extraction

Sample queries and outputs:

Q: 'অপরিচিতা' গল্পে কোন দ্বীপের উল্লেখ আছে?
Ans: আন্ডামান দ্বীপ

Q:কাকে অনুপমের ভাগ্য দেবতা বলা হয়েছে?
Ans:মামা

Q:অনুপমের বাবা কী করে জীবিকা নির্বাহ করতেন?

Ans:ওকালতি করে।

Q:অপরিচিতা গল্পের নারীর নাম কী?
Ans:কল্যাণী

Q:টাকার প্রতি আসক্তি কার?

Ans:মামার অস্থিমজ্জায় জড়িত।

Q: ট্রেনের স্টেশনে হতে কী খাওয়া কিনে নেয়?
Ans: চানা-মুঠ

Q:অপরিচিতা মেয়েটির সাথে কতোজন মেয়ে ছিল?

Ans:দুটি-তিনটি ছোটো মেয়ে।

Q: কল্যাণীর পিতার নাম কী?
Ans:শম্ভুনাথ সেন।

Q:What role did Harish play in the story?
Ans: Harish is a character who is trusted by the girl's father, Shastunathbabu, and is involved in the marriage arrangements.

Q: Who said "খাঁটি সোনা বটে" and about whom?
Ans:বিনুদাদার ভাষা সম্পর্কে বলা হয়েছে "খাঁটি সোনা বটে।"

Q:Who was Anupam's guardian and how did he influence Anupam's life decisions?
Ans: Anupam's guardian was his maternal uncle (mama). He influenced Anupam's life decisions by having a specific preference for a bride who would come from a humble background and not be wealthy, emphasizing that the girl should come with her head bowed.

Answer the following Questions:

Question: What method or library did you use to extract the text, and why? Did you face any formatting challenges with the PDF content?

Answer:

I used the multilingual_pdf2text library, specifically the PDF2Text class, to extract text from PDF files. This library is well-suited for handling multilingual documents, and since my PDF content was in **Bangla**, it provided better accuracy in extracting and preserving the original structure and language-specific characters. Additionally, it integrates with a structured Document model, making it easier to process the text further.

Yes, I did face some formatting challenges, such as inconsistent spacing and line breaks in the extracted Bangla text, especially around headings or multi-column layouts like table. To handle this, I applied post-processing steps like cleaning up unwanted newlines and merging fragmented lines to ensure smooth semantic chunking and embedding downstream.

Question: What chunking strategy did you choose (e.g. paragraph-based, sentence-based, character limit)? Why do you think it works well for semantic retrieval?

Answer:
I chose a **paragraph-based chunking strategy** because paragraphs naturally group related sentences, preserving context and meaning better than sentence-level or fixed-length splits. This helps the semantic retrieval system understand and match user queries with more coherent and meaningful text chunks.

Question: What embedding model did you use? Why did you choose it? How does it capture the meaning of the text?

Answer: I used `OpenAIEmbeddings` with the `"text-embedding-3-large"` model. I chose it because it provides **high-quality semantic embeddings**, capturing deeper context and meaning across languages. It works by representing text as vectors where semantically similar texts are placed closer together in vector space, enabling more accurate retrieval.

Question: How are you comparing the query with your stored chunks? Why did you choose this similarity method and storage setup?

Answer:

I'm comparing the query with stored chunks using **L2 (Euclidean) distance** via **FAISS**. I initialized the FAISS index with `IndexFlatL2`, which measures the distance between the query embedding and stored chunk embeddings. I chose this method for its simplicity and efficiency, especially with high-dimensional embeddings like `text-embedding-3-large`.

For storage, I'm using `PersistentDocstore` with `DocumentLocalFileStore` to ensure documents persist across sessions. The combination of FAISS for fast retrieval and local docstore for persistent storage provides a good balance between **speed**, **semantic accuracy**, and **scalability**.

Question: How do you ensure that the question and the document chunks are compared meaningfully? What would happen if the query is vague or missing context?

Answer: To ensure meaningful comparison, both the query and document chunks are embedded using `text-embedding-3-large` and compared using L2 distance in FAISS. This captures semantic similarity even if wording differs.

If the query is vague or lacks context, we **retrieve relevant chunks from vector storage** and **append them to the query**. This enriched prompt provides context to the language model, helping it generate more accurate and meaningful responses.

Question: Do the results seem relevant? If not, what might improve them (e.g., better chunking, better embedding model, larger document?

Answer:

The results are **sometimes relevant**, but not always consistent. This often happens when chunks lack full context or when the model can't clearly identify who is speaking.

To improve relevance and understanding:

- We can **summarize each chunk** and store the summary in the vector database to better capture its core meaning.
- A **hybrid retriever**—combining dense vector search with keyword-based methods like **BM25**—can improve both **precision** and **recall**.
- Including a **summary or gist of the full story** as metadata or prepending it to the prompt helps the language model understand the **broader narrative**.
- We can apply **Named Entity Recognition (NER)** to extract **person names** and store them as metadata, so the model better understands **who said what**.
- Additionally, using a **parent–child chunking system** allows us to store **smaller "child" chunks for fine-grained retrieval**, while linking them to **larger "parent" chunks** for full context. This helps the model retrieve specific details without losing the surrounding narrative.