{SAMPLE SET}

Gen AI Engineer / Machine Learning Engineer Assignment

Part 1: Retrieval-Augmented Generation (RAG) Model for QA Bot

Problem Statement:

Develop a Retrieval-Augmented Generation (RAG) model for a Question Answering (QA) bot for a business. Use a vector database like **Pinecone DB** and a generative model like **Cohere API** (or any other available alternative). The QA bot should be able to retrieve relevant information from a dataset and generate coherent answers.

Task Requirements:

1. Implement a **RAG-based model** that can handle questions related to a provided document or dataset.
2. Use a vector database (such as **Pinecone**) to store and retrieve document embeddings efficiently.
3. Test the model with several queries and show how well it retrieves and generates accurate answers from the document.

Deliverables:

- A Colab notebook demonstrating the entire pipeline, from data loading to question answering.
- Documentation explaining the model architecture, approach to retrieval, and how generative responses are created.
- Provide several example queries and the corresponding outputs.

Part 2: Interactive QA Bot Interface

Problem Statement:

Develop an interactive interface for the QA bot from Part 1, allowing users to input queries and retrieve answers in real time. The interface should enable users to upload documents and ask questions based on the content of the uploaded document.

Task Requirements:

1. Build a simple **frontend interface** using **Streamlit** or **Gradio**, allowing users to upload PDF documents and ask questions.
2. Integrate the backend from Part 1 to process the PDF, store document embeddings, and provide real-time answers to user queries.
3. Ensure that the system can handle multiple queries efficiently and provide accurate, contextually relevant responses.
4. Allow users to see the retrieved document segments alongside the generated answer.

Deliverables:

- A deployed QA bot with a frontend interface where users can upload documents and interact with the bot.
- Documentation on how the user can upload files, ask questions, and view the bot's responses.
- Example interactions demonstrating the bot's capabilities.

Guidelines:

- Use **Docker** to containerize the application for easy deployment.
- Ensure the system can handle large documents and multiple queries without significant performance drops.
- Share the code, deployment instructions, and the final working model through GitHub.

General Guidelines:

1. Ensure modular and scalable code following best practices for both frontend and backend development.
2. Document your approach thoroughly, explaining your decisions, challenges faced, and solutions.
3. Provide a detailed ReadMe file in your GitHub repository, including setup and usage instructions.
4. Submissions should include:
   - Source code for both the notebook and the interface.
   - A fully functional Colab notebook.
   - Documentation on the pipeline and deployment instructions.