Technical Assignment - RAG-Powered Telegram Support Bot

This assignment is designed to evaluate your understanding of Retrieval-Augmented Generation (RAG) systems, large language models (LLMs), backend API development, and optimization strategies in production-like environments.

Scenario

Your task is to build a lightweight RAG-based question-answering system accessible via a Telegram bot. The system should ingest documents, retrieve relevant context based on user input, and use an LLM (e.g., GPT-4, Gemini, or an open-source model like Mistral or LLaMA) to generate responses.

Technical Requirements

1. Backend: Use FastAPI to create a REST API with at least one route (`/ask`) that accepts a text question and returns an LLM-generated answer based on document context.

2. Document Handling: Load `.txt` documents from a local `./data/` folder. Index them using a vector search library such as FAISS or Chroma.

3. LLM Integration: Use any available LLM (OpenAI, Gemini via API, or Hugging Face models).

4. Telegram Integration: Connect your FastAPI backend to a Telegram bot that users can interact with.

5. Dockerization: Provide a Dockerfile to containerize your project. The image should expose the FastAPI service on port 8000.

6. Testing: Include at least one unit test (e.g., using `pytest`) to verify your `/ask` endpoint works as expected.

7. Deployment & Demo: You may use Google Colab to demo your pipeline. Gemini API is also allowed if available.

Performance Optimization Goals (150 words + reference each)

- Minimize LLM API usage (e.g., caching, pre-answering, filtering).

- Reduce response latency.

- Enable easy scaling for future deployment.

Expected Deliverables

- Git repository with source code.

- Optional: Google Colab notebook.

- Telegram bot demo (link/username).

- README file with setup, API usage, and architecture.

- Short report (`answers.md`) addressing:

- Why use RAG over standard LLM prompting?

- How did you handle latency?

- How would you scale this in production?

Section B: Personality Analysis from Text

1. Propose a non-LLM model (classic ML/statistics/tiny LM).

2. Write a prompt for a modern LLM (e.g., GPT-4 or Gemini).

3. Compare approaches and discuss hybrid possibility.

4. Implement in Google Colab (PyTorch/TF).

Section C: AI Assistant in Tax Organization

1. Propose a method to detect tax fraud (e.g., fake invoices).

2. What data would you analyze? What is the output?

3. How would LLMs help? Include logic/rules/analytics.

Google Colab implementation is highly encouraged.