Hackathon Task: Document-based RAG Chatbot

Objective

Build a chatbot capable of answering user queries based strictly on the content provided in a set of PDF and Word documents. The system must be built using a Retrieval-Augmented Generation (RAG) architecture and adhere to all the conditions listed below.

Scope of Work

You will be provided with a set of PDF and Word documents. Your task is to implement a chatbot that uses these documents as its only source of information to answer user queries.

General Conditions

1. The total time to process and respond to a user query must not exceed 15 seconds. (Applicable only when you are running on GPU)
2. LangChain or any component of LangChain is not allowed.
3. Only open-source LLMs and embedding models are to be used.
4. API calls to external models or services (e.g., OpenAI, Claude, Gemini, Mistral API, etc.) are not allowed.
5. The chatbot must accurately identify and display the exact document source used for each answer. (Must display document filename, page number, and exact chunk/section identifier used for the answer)
6. The displayed document link must correspond only to the content used in generating the answer, not all retrieved chunks. (Answer Generator LLM must take care of document link)
7. You may use any open-source framework for chunking the documents.
8. The chunking method you use must be justified with clear reasoning.
9. Only one vector database query is allowed per user question. (you can't query the DB multiple times)
10. You are allowed to use metadata-based filtering strategies if needed.
11. Qdrant must be used as the vector database (Nothing else).
12. The chatbot's answers must be directly extracted from the retrieved context.
13. If the LLM cannot find the answer in the provided context, it should tell the same to the user.

14. The complete solution must be able to run on a Tesla T4 GPU. (Must include GPU/CPU memory usage analysis and optimization strategies. If Tesla T4 is not available, provide detailed resource usage documentation showing compatibility with 16GB GPU memory limit)
15. You may use any platform to run your code (e.g., Kaggle, Google Colab, or your local system).
16. Create a UI using any of these (Streamlit / Gradio / Chainlit / Reflex) to solve this problem. Except for the mentioned frameworks, do not use anything else to create a UI.
17. Once completed, this entire solution should run on a local system without internet. (Assuming all models, dependencies, and data must be downloaded and cached locally.)

Optional Features (Not Mandatory)

1. You are not required to support greeting or casual conversation inputs.
2. You are not required to support follow-up or threaded questions.
3. You are not required to create a good interactive UI for this. A basic UI which can take user input text and return a response is enough.

Submission Requirements

1. Submit your complete codebase in a GitHub repository with a README file (MUST be a formatted README).
2. Content for README.md -> explaining setup, usage, and architectural decisions, observations, chunking strategy, retrieval approach, and hardware usage. (Order does not matter)
3. Submit a document which includes 10 (Must be 10) queries with response text and screenshot of the response in a FORMATTED document (PDF / Word / txt — anything on any shareable platform).
4. If you fail at any point of this task, MUST explain in the README why you failed, how long it would take to make this chatbot work, and how you would make it work.
5. You MUST submit a DEMO VIDEO LINK (YouTube, Google Drive, or any shareable platform) that satisfies all the following conditions:
    a. The video must show you (the presenter) on camera with your face clearly visible throughout the video.
    b. The video must include live testing of the chatbot on at least 5 questions taken from the 10 questions you will submit in your response document.

c. You are required to run the full application live during the recording. This means run the chatbot application from scratch in front of the camera.
d. Input the questions manually and show the real-time responses.
e. No part of the video should be cropped, cut, or edited. We want to see the entire interaction and system behaviour in real time. Pre-recorded interactions, UI mock-ups, or trimmed demos will lead to disqualification.

Bonus Points (Only do this if you have time left)

1. Create an architecture of this entire chatbot for production (Be creative).
2. Tell us in which part of the code you used AI (ChatGPT, Claude, Cursor, Windsurf...) to write code and how you generally use these kinds of tools.

Evaluation Criteria

Criteria	**Weight**
Accuracy of Answers	30%
Accuracy of Source Document Link	20%
Justification of Chunking Method	10%
Response Time Within Limit	10%
Code Quality and Modularity	10%
Extra Enhancements	10%
Good README file on GitHub	10%