Use Case Story: Building a RAG (Retrieval-Augmented Generation) Chatbot

Objective:

Build a **Retrieval-Augmented Generation (RAG) Chatbot** that allows users to interact with documents (PDF, audio, video) via a simple web UI and get responses with proper source references.

Use Case Overview:

1. The user accesses a **Streamlit-based UI**.
2. The user uploads a **PDF, Audio, or Video file**.
3. The file is processed and indexed into a vector store.
4. The user can then chat with the bot by asking questions related to the uploaded files.
5. The chatbot generates a relevant response using an **open-source LLM model** and shows the source reference.

Functional Requirements:

1️⃣ User Interface (Streamlit):

- Simple and clean UI where the user can:
    - Upload a file (PDF, Audio, Video).
    - Enter a question in a chat window.
    - See the chatbot's response along with source reference.

2️⃣ Backend API (FastAPI):

- Expose endpoints for:
    - File upload (PDF, Audio, Video).
    - Query handling.

- Implement middleware to:
    - Log incoming requests and responses.
    - Handle errors gracefully.
- Maintain basic logs (request timestamps, input size, user query).

3️⃣ *LLM Integration:*

- Use any open-source model (e.g., Hugging Face models).
- When a user asks a question:
    - Retrieve relevant context from the uploaded documents (using vector search).
    - Generate an answer based on the retrieved context.
    - Attach source reference in the answer.

4️⃣ *Docker Setup:*

- Provide a **Dockerfile** and **docker-compose.yml**.
- Ensure the entire app (Streamlit UI + FastAPI + LLM) runs in containers.
- Easy startup with a single command like:

```
docker-compose up –build
```

5️⃣ *Project Management with UV:*

- Use **UV (Universal Virtual environment)** for:
    - Dependency management (Python packages).
    - Environment isolation.
    - Project structure management.
    - Python version – 3.11

📑 Example User Flow:

1. Open the app in the browser.
2. Upload a PDF document (e.g., product manual).

3. Ask: “What are the safety precautions for using this device?”
4. Chatbot responds:

“the device must not be used near water due to electrical risks.

Reference : Page 1 – pdf name”

Non-Functional Requirements:

- Easy-to-use UI.
- Lightweight and modular backend structure.
- Fast response time (within seconds).
- Reproducible environment via Docker.

Goal for Interns:

- Design the Streamlit frontend to upload files and interact with the chatbot.
- Build a robust FastAPI backend with middleware and logging.
- Integrate an open-source LLM model for generating responses.
- Implement vector-based retrieval from uploaded files.
- Containerize the entire app using Docker.