RAG Chatbot with Groq, FastAPI, and Chainlit

# Overview

This project is a Retrieval-Augmented Generation (RAG) chatbot that allows users to upload PDFs and ask questions based on their contents. It uses Groq's LLM, Chainlit for UI, and FastAPI for backend processing. DuckDuckGo search is used for fallback web retrieval.

# Tech Stack

- LLM: Groq API (mixtral-8x7b-32768)  
- Frontend: Chainlit (chainlit==0.6.0)  
- Backend: FastAPI (fastapi==0.97.0)  
- Search: DuckDuckGo  
- Containerization: Docker + Docker Compose  
- Environment Handling: .env with python-dotenv

# Features

- Upload PDFs and interact with them  
- Automatically decomposes questions into sub-queries  
- Retrieves info from PDFs or web (DuckDuckGo)  
- Synthesizes answers using Groq's LLM  
- Simple frontend via Chainlit

# Project Structure

chatbot/  
├── app/ # FastAPI backend logic  
│ ├── main.py # API endpoints  
│ ├── rag.py # Query decomposition & retrieval logic  
│ ├── utils.py # PDF processing helpers  
│ ├── groq\_client.py # LLM API calls  
│ └── \_\_init\_\_.py  
├── chainlit\_app/  
│ └── app.py # Chainlit UI  
├── Dockerfile  
├── docker-compose.yml  
├── requirements.txt  
└── .env

# How It Works

1. User uploads PDFs → Text is extracted and stored in memory.  
2. User sends a question → It's decomposed into sub-questions.  
3. Each sub-question is searched in:  
 - PDF content (local vector store)  
 - Web (via DuckDuckGo) if not found locally  
4. The results are synthesized into a final answer using Groq's LLM.  
5. Response is shown in Chainlit UI.

# Running the Project

1. Clone and prepare:  
 git clone <your-repo-url>  
 cd chatbot  
  
2. Add environment variables in `.env` file:  
 GROQ\_API\_KEY=your\_groq\_api\_key  
 SEARCH\_API=duckduckgo  
  
3. Build and run using Docker:  
 docker-compose up --build  
  
- Chainlit UI: http://localhost:8001  
- FastAPI: http://localhost:8000

# Future Improvements

- Replace dummy PDF parser with pdfplumber or PyMuPDF  
- Add persistent vector store (e.g., FAISS or Chroma)  
- Add session-based history