FullStack RAG Chatbot Documentation

This document provides a complete guide to setting up, running, and using the FullStack RAG Chatbot.

1. Overview

This project is a complete **Retrieval-Augmented Generation (RAG)** system that allows users to chat with their own documents. They can upload PDF files or provide web page URLs, and the AI will answer questions based on the content of those sources.

The entire system runs locally on the machine, ensuring the user's data remains private(for now).

Key Technologies:

- Backend API: FastAPI
- Frontend Web App: Streamlit
- AI Orchestration: Langchain
- Local LLM & Embeddings: Ollama
- Vector Database: ChromaDB

---

2. Architecture at a Glance

The system uses a simple but powerful architecture. The frontend (web app) talks to the backend (API), which handles all the complex AI tasks.

User Flow:

Generated code

```
    [User] -> [Streamlit Web App] -> [FastAPI Backend] -> [AI Agents & Database] -> [Ollama LLM]
```

---

3. Getting Started: Installation

Prerequisites

- Python 3.8+: Make sure you have a modern version of Python installed.

- Ollama: You must have Ollama installed and running. Download it from https://ollama.com/.

Step 1: Get the AI Models

Once Ollama is running, open your terminal and pull the necessary models. This project uses mistral for chatting and mxbai-embed-large for understanding document content.

- Chatting model : Mistral model
- Embedding model : mxbai-embed-large

Step 2: Set Up the Project Structure

```
fullstack-rag-project/
├── app/
│   ├── __init__.py
│   ├── config.py          # Configuration settings
│   ├── vector_store.py     # Handles ChromaDB interactions
│   ├── document_processor.py # PDF processing logic ("Agent")
│   ├── web_scraper.py      # Web scraping logic ("Agent")
│   ├── rag_orchestrator.py # Core RAG chain logic
│   └── main.py             # FastAPI application
└── requirements.txt
```

---

4. How to Run the Application

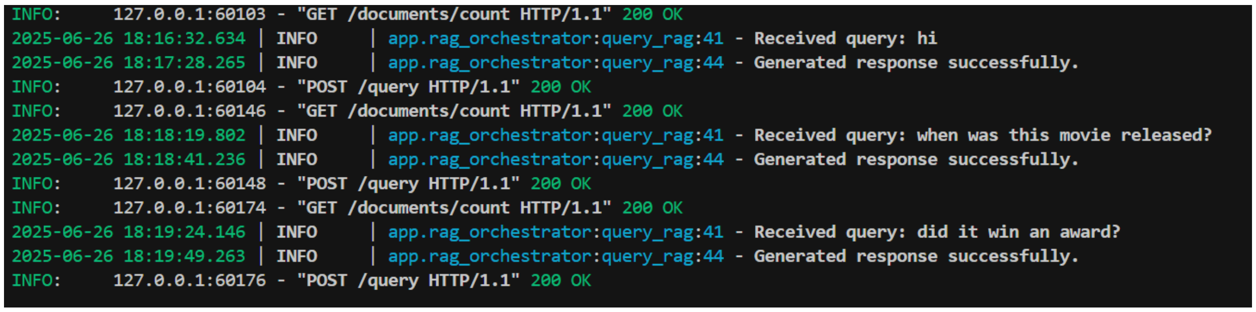
The system has two parts (backend and frontend) that need to be run at the same time. We need two separate terminal windows for this.

Terminal 1: Start the Backend (API)

In your first terminal, make sure your virtual environment is activated, then run the following command to start the FastAPI server:

Generated bash

uvicorn app.main:app --reload



```
INFO:     127.0.0.1:60103 - "GET /documents/count HTTP/1.1" 200 OK
2025-06-26 18:16:32.634 | INFO     | app.rag_orchestrator:query_rag:41 - Received query: hi
2025-06-26 18:17:28.265 | INFO     | app.rag_orchestrator:query_rag:44 - Generated response successfully.
INFO:     127.0.0.1:60104 - "POST /query HTTP/1.1" 200 OK
INFO:     127.0.0.1:60146 - "GET /documents/count HTTP/1.1" 200 OK
2025-06-26 18:18:19.802 | INFO     | app.rag_orchestrator:query_rag:41 - Received query: when was this movie released?
2025-06-26 18:18:41.236 | INFO     | app.rag_orchestrator:query_rag:44 - Generated response successfully.
INFO:     127.0.0.1:60148 - "POST /query HTTP/1.1" 200 OK
INFO:     127.0.0.1:60174 - "GET /documents/count HTTP/1.1" 200 OK
2025-06-26 18:19:24.146 | INFO     | app.rag_orchestrator:query_rag:41 - Received query: did it win an award?
2025-06-26 18:19:49.263 | INFO     | app.rag_orchestrator:query_rag:44 - Generated response successfully.
INFO:     127.0.0.1:60176 - "POST /query HTTP/1.1" 200 OK
```

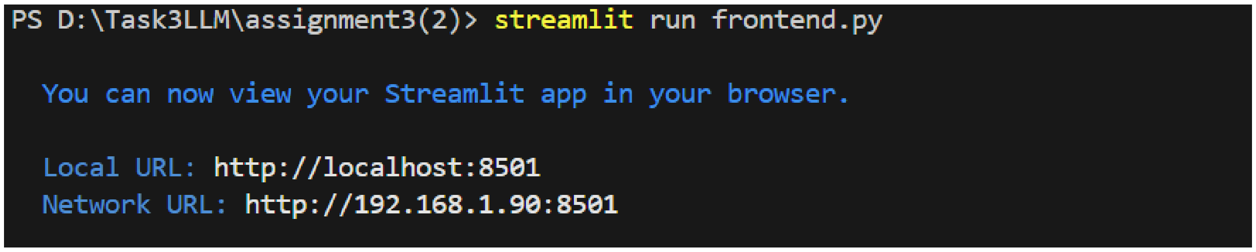
You will see a message indicating the server is running on http://127.0.0.1:8000. Leave this terminal running.

Terminal 2: Start the Frontend (Web App)

In your second terminal, activate the virtual environment again, then run this command to launch the Streamlit web application:

Generated bash

streamlit run frontend.py



```
PS D:\Task3LLM\assignment3(2)> streamlit run frontend.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.90:8501
```

This will automatically open a new tab in your browser. If it doesn't, navigate to http://localhost:8501.

---

5. How to Use the Web App

The web interface is designed to be simple and intuitive.

1. Add Your Content (Sidebar)
   - To upload a PDF: Click "Browse files" and select a PDF document from your computer. The system will process it automatically.
   - To add a website: Paste a URL into the "Or add a Web URL" text box and click the "Process URL" button.
2. Ask a Question (Main Window)
   - Type any question about the documents you've added into the chat box at the bottom of the screen and press Enter.
   - The AI will "think" for a moment and then provide an answer based on the information in your documents.
3. Manage Your Data (Sidebar)
   - The "Documents in DB" counter shows how many chunks of text are currently stored.
   - Click the "Clear All Documents" button to erase everything from the database and start fresh.

---

6. Project File Structure

For those interested in the code, here is a breakdown of the key files:

- frontend.py: The Streamlit web interface code.
- app/main.py: The FastAPI backend, defining all API endpoints (/upload, /query, etc.).
- app/rag_orchestrator.py: The core RAG logic that connects the retriever and the LLM.
- app/document_processor.py: The "agent" responsible for handling PDFs.
- app/web_scraper.py: The "agent" responsible for fetching web page content.
- app/vector_store.py: Manages all interactions with the ChromaDB vector database.
- app/config.py: Central configuration for model names, paths, etc.