**Design Document – Local RAG Chatbot with Ollama + PDFs**

**1. Overview**

This project implements a **Retrieval-Augmented Generation (RAG) chatbot** that allows users to ask natural language questions about the content of PDF documents.  
It runs **entirely locally** using:

* **Ollama** (for embeddings + LLM responses)
* **ChromaDB** (vector database for document retrieval)
* **Streamlit** (UI to interact with the chatbot)

The chatbot reads PDFs, converts them into vector embeddings, stores them in a vector database, and uses the embeddings to provide context-aware answers.

**2. Architecture**

**Components**

1. **PDF Loader**
   * Reads PDF files from a folder (project\_docs).
   * Extracts raw text using PyPDF.
2. **Embedding Generator**
   * Uses Ollama’s nomic-embed-text model to convert text into dense vector embeddings.
   * Each embedding represents the semantic meaning of a document.
3. **Vector Database (ChromaDB)**
   * Stores embeddings and corresponding document text.
   * Supports similarity search for retrieving relevant chunks when a question is asked.
   * Persistent mode ensures the database survives app restarts (chroma\_db/ folder).
4. **Query Processor**
   * When a user asks a question:
     + Converts the query into an embedding (via Ollama).
     + Retrieves top n most relevant documents from ChromaDB.
     + Passes these documents + question as a prompt to Ollama’s LLM (llama3.1:8b).
5. **LLM Answer Generator**
   * Ollama LLM uses the retrieved context to generate a natural-language answer.
   * Helps the model remain factual and grounded in the provided PDFs.
6. **User Interface**
   * Built with Streamlit.
   * Features:
     + Displays status of loaded PDFs.
     + Input field for user questions.
     + Full chat history (Q&A between **You** and **Bot**).
     + Clean, icon-free design for readability.

**3. Workflow**

**Step 1: Load Documents**

* PDFs placed in project\_docs/.
* Extracted text → embedded with Ollama → stored in ChromaDB.

**Step 2: User Asks Question**

* Input collected via Streamlit text box.
* Converted to embedding using Ollama.
* Similar documents retrieved from ChromaDB.

**Step 3: Generate Answer**

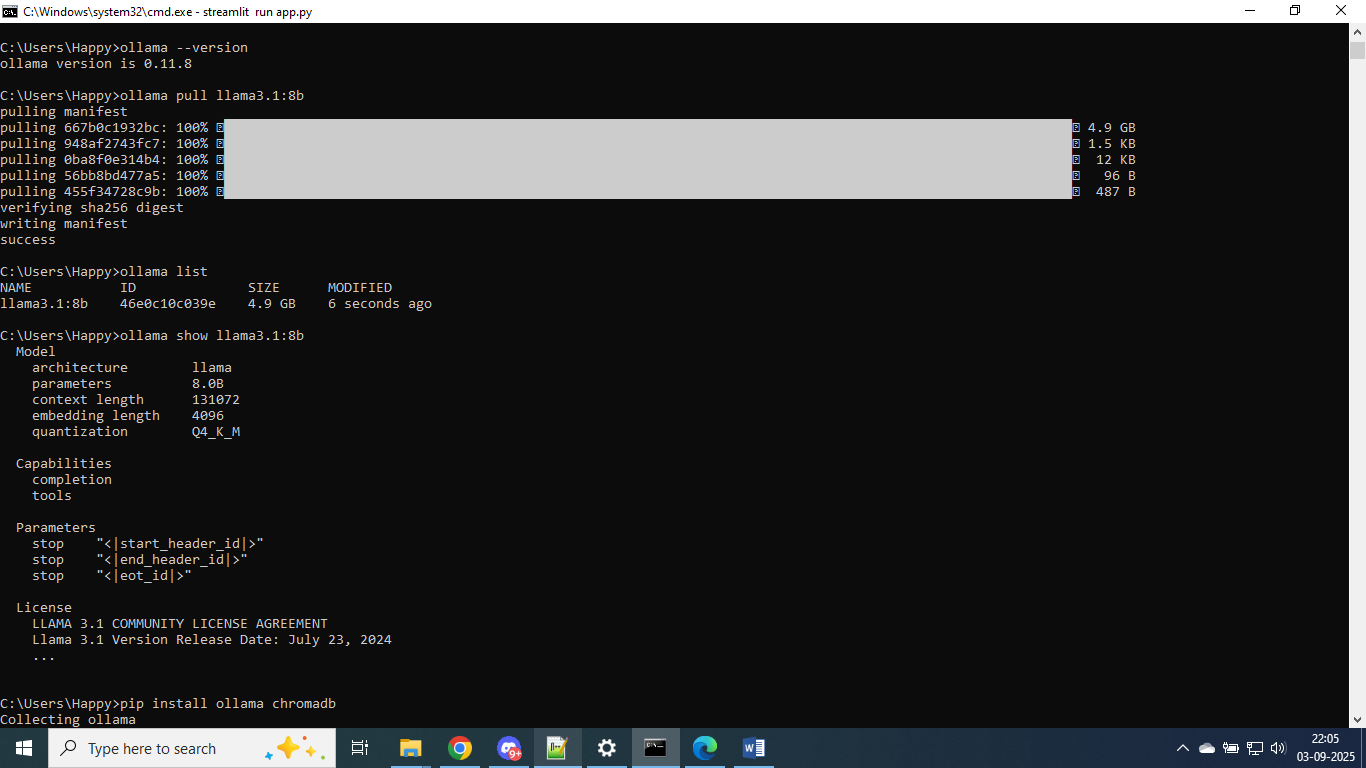
* Retrieved context + question passed to Ollama LLM.
* LLM generates context-aware answer.
* Answer displayed in UI alongside user’s query.

**Step 4: Maintain Conversation**

* All interactions stored in st.session\_state.chat\_history.
* Chat history is displayed below new inputs.

**4. Tech Stack**

| **Component** | **Tool / Library** |
| --- | --- |
| LLM | Ollama (llama3.1:8b) |
| Embeddings | Ollama (nomic-embed-text) |
| Vector Database | ChromaDB (PersistentClient) |
| Document Loader | PyPDF |
| UI | Streamlit |
| Language | Python 3.13 |



To run this project  
  
“streamlit run app.py”

