Offline Retrieval-Augmented Generation (RAG) Chatbot

1. Project Overview

This project uses a **fully offline RAG chatbot** that answers user questions based on a custom set of documents.
The system runs **on a local machine** using **open-source models**.

---

2. Architecture

The system consists of three main components:

i. **Ingest.py:** Loading documents and chunking the texts.

ii. **Rag_pipeline.py:** Embedding formation and responses on user's query.

iii. **App.py**: Streamlit App, handles all the steps.

---

3. Models and Tools Used

a. Embedding Model: SentenceTransformer (all-MiniLM-L6-v2)

b. Vector Database: FAISS

c. Language Model: Orca Mini 3B (GGUF, quantized). It is loaded using GPT4all and runs offline on CPU with low memory usage

d. Frameworks:
- Python
- Streamlit
- SentenceTransformers
- FAISS
- GPT4All
- PyPDF

---

4. Installation and Execution

1. **Install dependencies:**

   pip install -r requirements.txt

2. **Download LLM model:**

   - Download Orca Mini GGUF model
   - Place it inside the models/ directory

3. **Add documents**
   - Upload PDFs into data/docs/
   - Or Upload via the Streamlit UI

4. **Build a knowledge base**
   python ingest.py

   or click **“Rebuild Knowledge Base”** in the UI

5. **Run the application**
   streamlit run app.py

---

5. Usage Instructions

- Upload documents (only once unless documents change)
- Click **Rebuild Knowledge Base**
- Ask questions through the chatbot interface
- The system retrieves relevant context and generates answers

---

7. Conclusion

This project demonstrates a **complete offline RAG system** built using open-source tools.
I received a practical understanding of document ingestion, vector retrieval, and context-grounded answer generation, suitable for real-world deployment.