Python RAG Pipeline with Streamlit + LangChain

🎯 Objective

Build a Retrieval-Augmented Generation (RAG) chatbot using Python that:
Ingests custom .txt content (e.g., blog posts
Embeds and indexes it using FAISS
Uses LangChain to retrieve relevant chunks
Uses a Hugging Face model (GPT2 or Mistral) to answer querie
Includes a basic Streamlit UI for interaction

✅ Deliverables

A Colab-compatible .ipynb notebook
A Streamlit app script (app.py) to run the chatbot locally
Sample output from 2 questions
Link to GitHub repo or Google Drive (if required)

📁 Project Structure

```
rag_test/
├── blogs/
│   ├── blog1.txt
│   └── blog2.txt
├── rag_pipeline.ipynb      # Main notebook for logic
├── app.py                  # Streamlit UI
└── requirements.txt        # Dependencies
```

🧠 Tools to Use

LangChain – Pipeline orchestration
sentence-transformers – Embeddings (all-MiniLM-L6-v2 or bge-small)
FAISS – Vector store
transformers – LLM
Streamlit – Frontend UI

📌 Task Breakdown

Step 1: Data Ingestion & Preprocessing
Load text from blogs/ folder
Chunk text into ~200-word sections with titles preserved
Clean basic punctuation and newlines

Step 2: Embedding + Vector Store

Use sentence-transformers to generate embeddings
Store in FAISS with document metadata
Save and reload FAISS index for persistence

Step 3: LangChain Integration

Create a FAISS Retriever with LangChain
Use ConversationalRetrievalChain with:
A basic prompt template
A LLM like GPT2 or any small Hugging Face model

Example queries:

“What is energy mastery?”
“How do high performers avoid burnout?”