AI Chatbot with RAG Architecture – Technical Report
By: Niharika Bhatia

1. Document Structure and Chunking Logic

In the original full RAG pipeline, PDF documents are first loaded using PyPDFLoader. The text is then split into smaller, overlapping chunks using RecursiveCharacterTextSplitter. This is done to preserve contextual integrity while keeping chunks small enough for the embedding model to process efficiently.

- **Chunk Size:** 500 characters
- **Chunk Overlap:** 50 characters
- **Purpose:** Avoid loss of context across page or paragraph boundaries.

Note: In the current demo version, PDFs are not used. Instead, we simulate document responses using fixed logic to illustrate core chatbot behavior.

2. Embedding Model and Vector Store

In the full version:

- **Embedding Model:**
  HuggingFaceEmbeddings(model_name="BAAI/bge-small-en-v1.5")
  This model converts each document chunk into high-dimensional

vectors for semantic retrieval.

- **Vector Store:**
  FAISS is used as the vector database for storing and performing nearest-neighbor searches over the document embeddings.

  FAISS allows fast similarity search and is suitable for local/offline retrieval pipelines.

3. Prompt Format and Generation Logic

In a complete RAG pipeline, a user's query is embedded and compared to the stored document vectors. The top-k most relevant chunks are retrieved and used as context for the LLM. The prompt structure is:

```
You are a legal assistant. Based on the following context, answer the question accurately

Context:
<contextual chunks>

Question:
<user query>

Answer:
```

In the **basic version (no vector DB)**, a simple hardcoded dictionary or keyword-matching function simulates responses.

4. Example Queries and Responses

Query	Response	Notes
“What is an NDA?”	“An NDA, or Non-Disclosure Agreement, is a legal contract...”	✅ Expected behavior
“When does the contract expire?”	“The contract expires 12 months from the effective date.”	✅ Simulated logic
“What are the arbitration terms?”	“Arbitration will be conducted in New Delhi under Indian law.”	✅ Simulated
“Can I use this template in Germany?”	“This contract is governed by Indian law. Use in other jurisdictions should be reviewed by local counsel.”	✅ Caveat response
“What is the president’s name?”	“I’m not sure. This is outside the scope of the document.”	❌ Hallucination prevented

5. Notes on Hallucinations, Limitations & Latency

- **Hallucinations:** In the hardcoded demo, no hallucinations occur. In full RAG mode, hallucinations may appear when:
    - Retrieval returns irrelevant chunks.
    - Model over-generalizes without context.
- **Limitations:**

    - FAISS does not scale well to huge datasets.
    - BGE-small model is fast but lacks deep reasoning.
    - Streamlit does not support full chat history by default.
- **Latency Factors:**
    - Embedding and retrieval are fast (milliseconds).
    - LLM generation speed depends on backend (OpenAI API, local model, etc).

Conclusion

This basic prototype simulates a Retrieval-Augmented Generation (RAG) chatbot for document Q&A. Future enhancements include:

- Real PDF ingestion
- Chat history tracking
- Feedback loop for improving answer quality