RAG Chatbot Project Report

1. Document Structure and Chunking Logic

The PDF document (AI Training Document.pdf) is first parsed using PyPDF2. Text is extracted from each page and cleaned using regular expressions. Once the raw text is obtained, it is split into smaller overlapping chunks. We use LangChain's RecursiveCharacterTextSplitter with a chunk size of 250 words (approximately 1500 characters) and overlap of 50 words (approximately 300 characters). This ensures that context is preserved across chunk boundaries.

2. Embedding Model and Vector DB

For embedding generation, we use SentenceTransformer's 'all-MiniLM-L6-v2' model. It produces 384-dimensional dense vectors optimized for semantic similarity tasks. These vectors are stored and indexed in a FAISS (Facebook AI Similarity Search) flat L2 index, allowing fast and accurate nearest-neighbor search for relevant chunks.

3. Prompt Format and Generation Logic

When a user submits a query, we retrieve the top relevant chunk(s) from FAISS using cosine similarity. The chunk(s) along with the user query are passed into the LLM prompt. The LLM (e.g., LLaMA 3 running locally via Ollama) is instructed to answer the query strictly based on the retrieved content. Responses are streamed to the user in real-time via Streamlit.

4. Notes on Hallucinations, Model Limitations, or Slow Response

- Hallucinations: The LLM may generate information not present in the source document if prompt control is weak.
- Limitations: FAISS does not handle full document ranking, and context limited to top-k chunks.
- Slow Response: Initial model load time (especially via Ollama) can cause delay; embedding generation can also be slow for large documents.