Technical Write-Up: Document Intelligence Assistant

---

1. Overview

The Document Intelligence Assistant is a full-stack GenAI-based application designed to semantically understand documents uploaded by users and answer natural language questions based on their content. It combines document preprocessing, embedding generation, vector-based semantic search, and large language model reasoning in a streamlined Retrieval-Augmented Generation (RAG) pipeline.

---

2. Embedding Model: HuggingFace `MiniLM-L6-v2`

- We chose `all-MiniLM-L6-v2` from HuggingFace due to its:
  - Lightweight size (~80MB) with excellent performance
  - Compatibility with local CPU/GPU inference
  - Proven performance in semantic search and sentence-level embeddings
- It enabled fast and meaningful chunk embeddings without requiring external APIs.

---

3. Chunking Strategy

- **Chunker Used**: `RecursiveCharacterTextSplitter`
- **Chunk Size**: 500 characters
- **Overlap**: 50 characters
- **Why Recursive?**
  - Maintains semantic cohesion within text blocks
  - Efficient fallback from paragraphs → sentences → characters

    - Prevents mid-sentence breaks and enhances embedding quality

---

4. Prompt Engineering Approach

- The retrieved chunks are inserted into a system-style prompt sent to Groq's LLM.

Prompt Template:

Use the following context to answer the user's question.

Context:

[retrieved chunks go here]

Question: [user input]

Answer:

-
- This structure ensures:
    - Grounded reasoning
    - No hallucinations
    - Context-sensitive answers

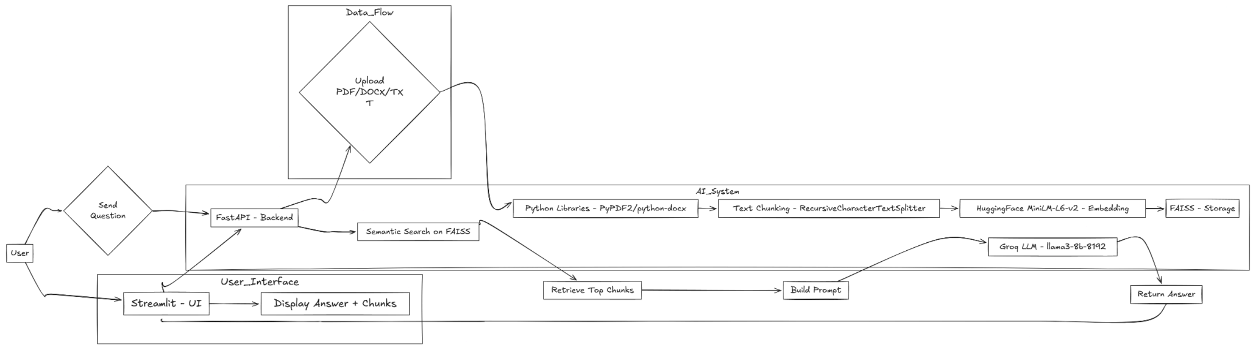
---

5. RAG Architecture Overview

- Users upload `.pdf`, `.docx`, or `.txt` files via Streamlit.
- Files are sent to the FastAPI backend where:
    - Text is extracted (using PyPDF2, python-docx, or file reading)
    - Text is chunked and embedded

    - Embeddings are stored in a FAISS vector store
- When a question is asked:
    - FAISS performs similarity search
    - Top relevant chunks are selected
    - Prompt is constructed and sent to Groq's `llama3-8b-8192`
    - Answer is returned and shown with context and confidence score

[System Architecture Diagram]



---

6. Challenges Faced & Solutions

🔴 Chroma Crash on Windows

- **Issue**: Persistent `ConnectionResetError 10054` with Chroma
- **Fix**: Switched to FAISS for local vector storage — lightweight and stable

🔴 HuggingFace Embedding Deprecation

- **Issue**: Warning for deprecated embedding class
- **Fix**: Switched to new `langchain-huggingface` interface

🔴 FastAPI → Streamlit disconnection

- **Issue**: Streamlit failed silently on file uploads
- **Fix**: Added response delay (`time.sleep(0.1)`) + removed `.persist()` usage

🔴 **User confusion on context**

- **Issue**: Lack of transparency on how answers are formed
- **Fix**: Exposed retrieved chunks in the frontend with simulated confidence score

🔴 **LLM hard fail without fallback**

- **Issue**: If Groq API call failed, entire chat broke
- **Fix**: Added fallback mechanism to return a graceful message from a dummy LLM or backup route

---

7. Outcome

- The app now supports real-time semantic Q&A over uploaded documents.
- Stable, testable, and API-free (except for Groq)
- Compatible across OS platforms due to FAISS choice
- Improved UX with confidence scores and fallback logic

---

8. Future Enhancements

- Multi-modal support (OCR-based image documents)
- Response streaming from LLM
- Conversation memory
- Document summarization feature

- Query rewriting to expand vague questions
- Multiple LLM fallback (OpenRouter, Gemini, etc.)

Submitted by: Aayush Dharpure
Project: GenAI Document Intelligence Assistant