Brief Technical Write-up

Tech Stack Choices

PDF Parsing	: PyPDF2 - Lightweight, easy to integrate.
Text Chunking	: LangChain RecursiveCharacterTextSplitter - Supports context overlap.
Embedding Model	: TF-IDF (Scikit-learn) - Fast, local, and API-free.
Vector Search	: FAISS - Efficient nearest-neighbor search for retrieval.
LLM (Cloud)	: OpenAI GPT-3.5 - High-quality structured response generation.
LLM (Offline)	: Ollama with LLaMA 3 - Fully offline generation, privacy-first.
Frontend UI	: Streamlit - Simple interactive web interface.

Response Structuring Approach

- Prompt engineering ensures the model only responds using retrieved document context.
- Structured formatting (bullet points, tables, paragraphs) improves clarity.
- Fallback prompt added: "This information is not available in the document." to prevent hallucinations.
- Top-K (default 3) most relevant chunks are passed to the LLM for generation.

Challenges Faced & Solutions Implemented

Challenge: sentence-transformers errors

Solution : Replaced with TF-IDF + FAISS for embedding and retrieval.

Challenge: OpenAI v1+ API compatibility

Solution : Updated to openai.ChatCompletion (v1.0+ syntax).

Challenge: Local-only functionality

Solution : Integrated Ollama to run LLaMA 3 locally via HTTP POST requests.

Challenge: PDF format inconsistency

Brief Technical Write-up

Solution : Used PyPDF2 with fallback logic for non-standard formats.

Challenge: Streamlit path errors on Windows

Solution : Used sys.path.append and restructured folders for relative imports.