High-Level Design (HLD) – PDF RAG System

Project Title: Retrieval-Augmented Generation (RAG) System for PDF Documents
Author: Anurag Mishra
Date: 2025-09-01

1. Project Overview

The goal of this project is to develop an end-to-end RAG system that can ingest PDF documents (native + scanned), perform retrieval-based question answering, generate answers with citations, and highlight exact evidence in the source pages. The system is fully local, using open-source tools like SentenceTransformers, ChromaDB, PyMuPDF, and LLaMA3.

2. System Architecture

Components:

1. PDF Ingestion
   - Input: Raw PDFs (data/raw_pdfs/)
   - Output: pages.jsonl (text and metadata per page)
   - Handles both native PDFs and scanned PDFs with OCR
2. Chunking
   - Splits text into manageable chunks (default: 1000 chars with 200 char overlap)
   - Output: chunks.jsonl with chunk IDs, page numbers, pdf_path
3. Embedding + Indexing
   - SentenceTransformer (all-MiniLM-L6-v2) generates embeddings per chunk
   - Embeddings stored in ChromaDB for fast similarity search
4. Retriever
   - Query embeddings compared against ChromaDB
   - Top-k most relevant chunks returned
5. LLM Query
   - Prompts constructed with retrieved chunks
   - LLaMA3 generates answer with citations [p.N]
6. Highlighting

- Evidence snippets highlighted in original PDF using PyMuPDF
- Output: Annotated PDF

7. Web UI (Optional)
   - Streamlit interface for selecting PDFs, asking queries, and downloading highlighted PDFs

3. Module Descriptions

Module	Responsibility	Input/Output
ingest.py	Extracts text + metadata from PDFs	Input: PDF, Output: pages.jsonl
chunk.py	Splits pages into chunks	Input: pages.jsonl, Output: chunks.jsonl
embed_index.py	Generates embeddings and builds ChromaDB index	Input: chunks.jsonl, Output: ChromaDB
rag_pipeline.py	Handles query, retrieval, LLM prompt, and evidence list	Input: query, Output: answer + evidences + annotated PDF
highlight.py	Highlights evidence text in PDF using PyMuPDF	Input: PDF + evidences, Output: annotated PDF
app.py	Streamlit UI for query and download	Input: user query, Output: UI + highlighted PDF

4. Data Flow

1. Ingest PDF -> extract pages -> store as JSONL
2. Chunking -> split pages into overlapping chunks -> store as JSONL
3. Embedding & Indexing -> encode chunks -> store in ChromaDB
4. Query -> user asks a question -> embed query -> retrieve top-k chunks -> construct prompt
5. LLM Answer Generation -> LLaMA3 generates answer using prompt
6. Highlight Evidence -> locate evidence in original PDF -> produce highlighted PDF
7. UI -> Streamlit displays answer, evidence snippets, download option

5. Additional Requirements / Notes

- Supports local-only execution (no cloud APIs required)
- Chunk size & top-k configurable for performance
- Handles scanned PDFs with OCR
- Optional Web UI allows demonstration for demo video

6. Deliverables Summary

Deliverable	Description
HLD Document	This PDF/DOCX with system overview and diagrams
Source Code	src/, scripts/, requirements.txt, README.md
Indexed Data	data/processed/ & ChromaDB (data/index/chroma)
Demo Video	Short screen recording showing end-to-end workflow