High-Level Design (HLD) — Local RAG with Citations

1) Objective:

Build a local, open-source Retrieval-Augmented Generation (RAG) system that ingests PDFs (native + scanned), retrieves relevant text, and produces answers with file + page citations and highlighted evidence.

2) Scope:

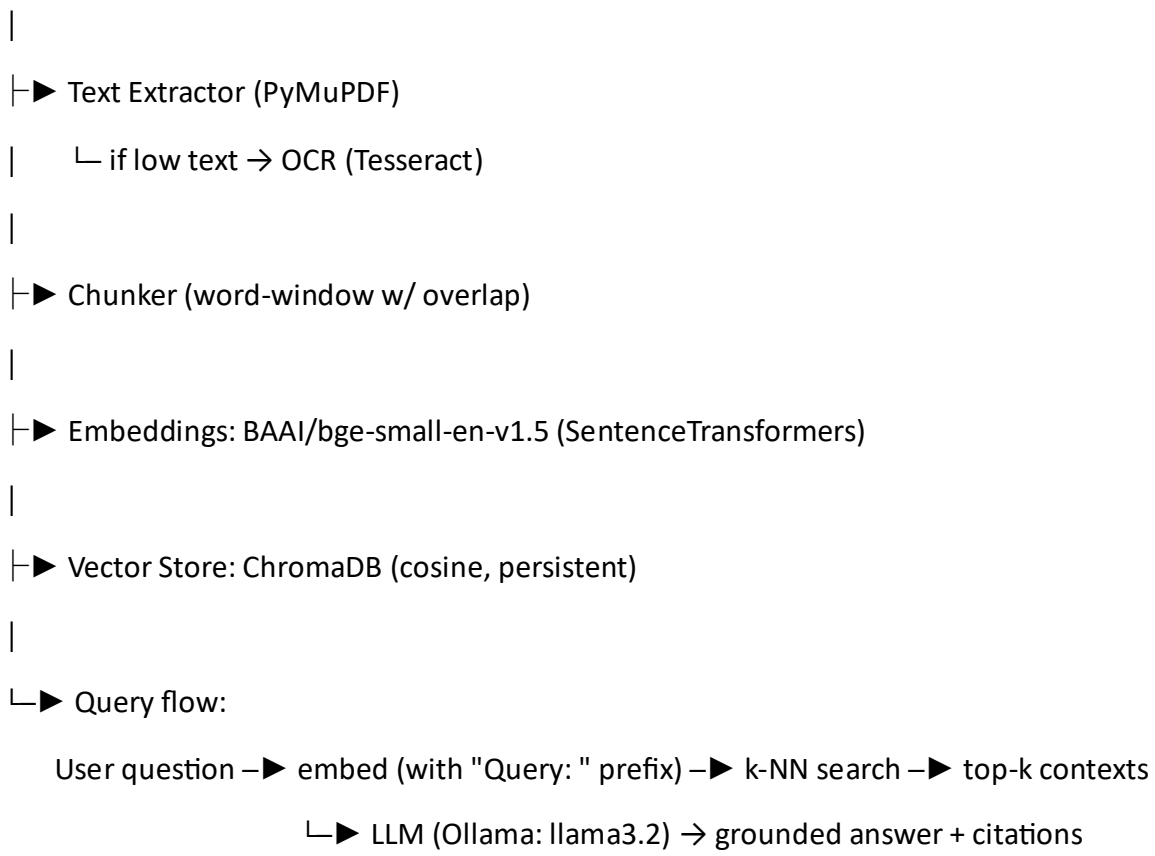
-In: Single/multiple PDFs, English text, scanned pages with OCR fallback, local inference.

-Out: CLI for ingest + ask, optional API/UI (de-scoped for submission), short demo video.

-Non-goals: Multi-user auth, cloud deployment, advanced eval dashboards.

3) Architecture (Overview):

PDF(s)



4) Data Flow:

1. Ingest
   PDF → per-page text (OCR if needed) → chunk into 800-word windows (120 overlap) → embed → add to Chroma with metadata.
2. Ask
   Question → embed (query-tuned) → top-k retrieval → build prompt with contexts → LLM generates answer constrained to context.
   Post-process: unique citations and bold highlight query terms in shown contexts.

5) Components:

-Extractor: PyMuPDF for native text; if len(text) < threshold, render page to image & run Tesseract OCR.

-Chunker: Fixed word windows to preserve locality; overlap avoids boundary loss.

-Embeddings: BAAI/bge-small-en-v1.5 (open, fast, strong for retrieval).

-Index: ChromaDB persistent collection (hnsw:space=cosine).

-Generator: Ollama with llama3.2, accessed via HTTP POST /api/generate.

-CLI:

ingest → builds/updates index from data/source_pdfs/

ask "<question>" → prints Answer, Citations, and Top contexts with highlighted evidence.

6) Key Design Choices:

-Local-only stack to fit privacy/offline constraints.

-OCR-on-demand to keep ingest fast.

-Query prefix "Query: " to match BGE query encoder expectations.

-Evidence visibility: show source/page + bolded terms in retrieved text to prove grounding.

-Timeout bump (HTTP) to tolerate first-run model warm-up.

7) Quality & Evaluation (measurable):

-Retrieval Hit@3: For 5–10 curated questions, check if the correct page appears in top-3.

-Answerability: If info absent from context, model responds: *"I don't know from the provided documents."*

-Latency: Log ingest time and avg ask latency on local machine.

-Manual spot-check: Verify citations correspond to the displayed evidence.

8) Assumptions & Risks:

-OCR errors on low-quality scans → mitigate by increasing DPI (e.g., 300) or adding language packs.

-Model quality: small local LLMs can be terse/vague; better prompts or a reranker can help.

-Hardware constraints: first generation may be slow; cached runs are faster.

-Document variety: heavy tables or images aren't fully parsed (out of scope).

9) How to Run (Windows):

```
# in project root
python -m venv .venv
.venv\Scripts\activate
pip install -r requirements.txt
ollama pull llama3.2

# add PDFs to: data\source_pdfs\
python app/rag_pipeline.py ingest
python app/rag_pipeline.py ask "What are the submission requirements?"
```

10) Demo Plan:

1. Show project tree briefly.
2. Run ingest (already indexed is fine; it shows counts).
3. Run 2–3 targeted questions (deadline, submission formats, how to submit).
4. Point out Citations (file & page) and bolded evidence in "Top contexts".

11) Future Improvements:

-Cross-encoder reranking (e.g., BGE reranker) to boost precision.

-Sentence-level evidence scoring & span highlighting.

-Chunking by layout (titles/sections) for semantically cleaner chunks.

-Lightweight FastAPI + Streamlit UI (already scaffold-ready).

-Automated eval (e.g., ragas) on a small QA set.

12) Submission Artifacts:

-HLD (this document) as PDF.

-Code: project folder with app/rag_pipeline.py, requirements.txt, README.md, data/source_pdfs/task.pdf.

-Video: short screen recording demonstrating ingest + ask + citations/evidence.