Minor Project work summary sheet for Quick reference of Panel and Supervisor

Project Title: LocalDocu: AI-Powered Document Assistant

Authors: Pranav Sharma, Kush Kansal, Prakhar Sighal

Date: November 21, 2025

1. Motivation behind the project:

The primary motivation is to address privacy, latency, and cost challenges associated with cloud-based AI services. LocalDocu aims to provide a secure, locally executable AI agent that prevents sensitive data exposure by localising inference and storage, eliminates per-query costs, and enables offline operation. Furthermore, it seeks to investigate the reliability of hierarchical RAG and multimodal retrieval when constrained to local consumer hardware.

2. Type of project: Development cum research project

The project combines engineering deliverables—frontend, backend, deployment packaging—with research aspects such as hierarchical Retrieval-Augmented Generation (RAG), reranking strategies, multimodal LLaVA integration, and structured citation enforcement.)

3. Critical Analysis (Gaps Addressed)

a. Gaps in Existing Systems

Most document-assistant tools depend on cloud-based AI, causing privacy concerns, latency, and operational costs. They use flat retrieval models, which are inefficient for large documents, and lack multi-modal support, making them unable to interpret images or visuals embedded in PDFs.

b. Gaps Addressed by LocalDocu

1. Privacy-Preserving Local AI: Runs all processing locally, ensuring data never leaves the user's device.
2. Hierarchical Retrieval Architecture: Combines summary-level and detailed-level vector stores with semantic chunking and reranking for precise, context-aware retrieval.
3. Multi-Modal Integration: Uses a vision-language model (LLaVA) to analyse images and embed visual information alongside text for unified query handling.
4. LocalDocu ensures data privacy, improves retrieval accuracy, and enables text–image

reasoning in a fully offline setup.

New Technologies & Tools Learned:

- Languages: Python 3.10+ (Backend), TypeScript, JavaScript (Node.js for Progress Service).
- Frameworks: FastAPI (Backend), Next.js (Frontend UI), LangChain
- AI/ML Tools: Ollama (Local Inference Server), ChromaDB (Local Persistent Vector Store), FlashRank (Reranking), LLaVA (Vision Model), HuggingFace Embeddings.

4. Overall design of project (Workflow & Data Model):

1. Ingestion: User uploads a PDF; the backend splits text into semantic chunks using LangChain's SemanticChunker and extracts images via PyMuPDF.
2. Image Processing: Extracted images are passed to the LLaVA model to generate textual summaries.
3. Storage: Summaries are stored in summary_store (coarse index), and detailed text/image chunks are stored in detailed_store (fine-grained index).
4. Retrieval: The system retrieves top matching document summaries first, then fetches and reranks detailed chunks from the detailed_store using FlashRank.

5. Features built, the Programming language used:

Key Features:

- Privacy-First Local Inference: The entire pipeline runs locally with no cloud data transmission.
- Hierarchical RAG: Two-level vector store (summary + detailed chunks) for efficient, context-rich retrieval.
- Multimodal Processing: Integrates LLaVA to extract and interpret images inside PDFs.
- Real-time Progress Tracking: Posts ingestion status to a separate microservice to prevent HTTP timeouts and provide visual feedback.

6. Proposed Methodology (Pipeline Details):

The project follows a specific pipeline methodology:

1. Ingestion: PDFs are loaded via PyMuPDFLoader; images are extracted and summarised using generate_image_summary with LLaVA.
2. Processing: Text is split using SemanticChunker to balance semantic coherence. A small model (Mistral) generates summaries for these chunks.
3. Indexing: Embeddings are generated using HuggingFaceEmbeddings. Document summaries

go to summary_store; chunks go to detailed_store.

4. Retrieval: query_rag implements hierarchical retrieval: it searches summaries first, extracts relevant doc_ids, and then retrieves specific chunks.
5. Reranking: Retrieved chunks are reranked using FlashRankRerank to select the top-K most relevant segments.

7. Algorithm/Description of the Work:

Core Algorithms:

1. Semantic Chunking: Splits text based on embedding similarity rather than fixed character counts to improve downstream LLM context quality.
2. Hierarchical Retrieval:
    - Step 1: summary_retriever.search(prompt) filters relevant documents.
    - Step 2: detailed_retriever fetches chunks only for those filtered documents.

8. Division of the work among students:

1. Kush Kansal: Architecture definition, backend orchestration, API definitions, overall system integration, and tests.
2. Prakhar Singhal: Frontend implementation (Next.js), design and UX, chat UI components, and progress UI.
3. Pranav Sharma: Ingestion pipeline, chunking & summarisation logic, image extraction, and LLaVA integration.

9. Results & Performance Evaluation:

- Quantitative Observations:
    - Ingestion Time: 10-50 page documents processed in 15-30 seconds; 100-page documents take 4-8 minutes.
    - Query Latency: Median generation latency is 4-6 seconds for typical document setups on an i5 CPU with 16GB RAM.
- Qualitative Observations:
    - Reranking: FlashRank improved contextual relevance in ~70% of ad-hoc test queries.

10. Conclusion

LocalDocu successfully demonstrates that a privacy-first, offline document assistant is viable using consumer-grade hardware. By combining hierarchical RAG with multimodal capabilities, the system solves key issues regarding data privacy and cloud costs. The modular architecture allows for future expansion into more complex file types and advanced verification stages, proving to be a robust foundation for secure document analysis.