System Design

Open GR --WM

Architecture Intent

A local-first Graph RAG product that ingests PDFs, routes extraction across text and vision models, builds a persistent knowledge graph, and serves grounded chat answers with source citations and quality checks.

Design Principles

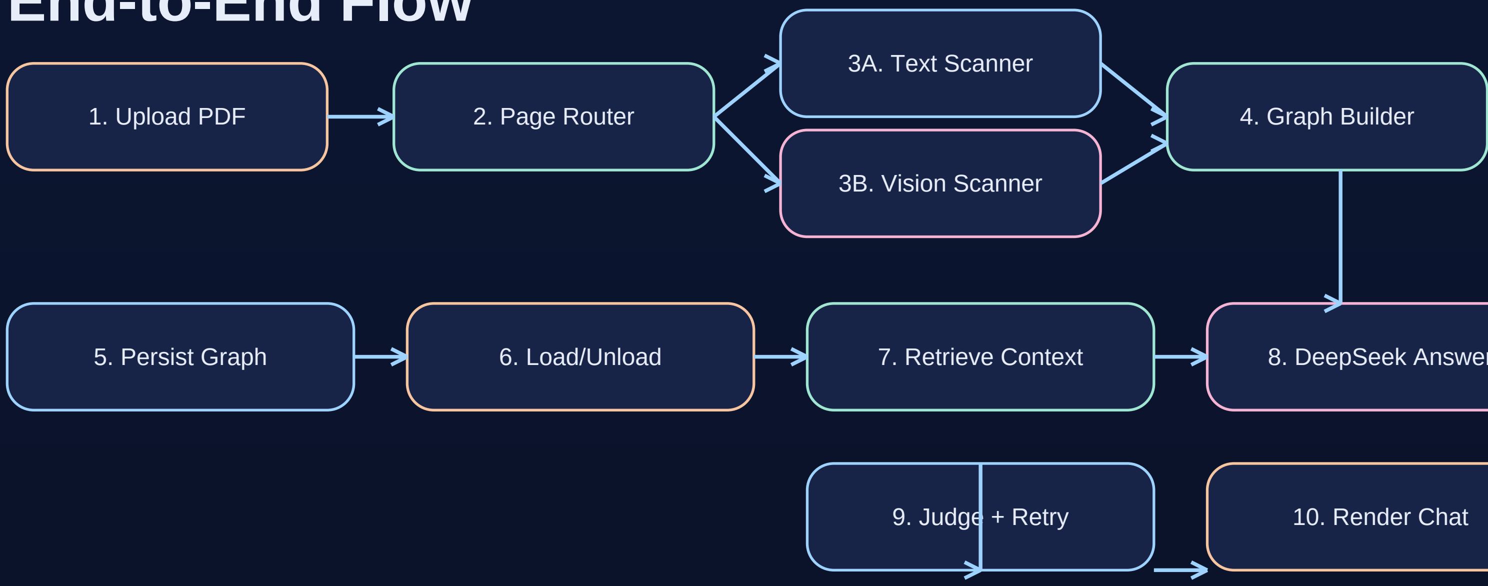
1) Fully local execution. 2) Model-specialized pipeline. 3) Persistent graph lifecycle. 4) Fast retrieval before deep reasoning. 5) Explainability via chunk/graph citations.

Model Allocation

llama3.2 handles fast text extraction + judge checks, llama3.2-vision handles visual pages, and deepseek-r1:14b handles rich chat reasoning.

Open GR --WM
ARCHITECTURE MAP

End-to-End Flow



1. Upload PDF

2. Page Router

3A. Text Scanner

3B. Vision Scanner

4. Graph Builder

5. Persist Graph

6. Load/Unload

7. Retrieve Context

8. DeepSeek Answe

9. Judge + Retry

10. Render Chat

Key Choice Callouts

Page-level routing avoids expensive vision inference on text-heavy pages. Chunk retrieval narrows context before reasoning. LLM-as-judge catches unsupported claims and triggers one retry. Storage supports load/unload/delete and multi-PDF graph augmentation.

Component Rationale

Ingestion Layer

PyPDF2 extracts text per page. Image-page detection and text-density checks enable selective vision inference.

Graph Extraction

NetworkX MultiDiGraph stores entities and predicates. JSON parsing is resilient to model filler and malformed blocks.

Embeddings & Retrieval

all-MiniLM-L6-v2 embeddings support fast cosine retrieval over chunks before LLM reasoning.

Reasoning & Quality

deepseek-r1:14b produces the answer, while a fast local judge model flags weak grounding and requests a retry.

Persistence Lifecycle

Graph, triples, chunks, embeddings, and metadata are saved per graph. Users can load, unload from RAM, augment, or delete.

Operational Playbook

Performance Controls

Tune build mode (Fast/Balanced/Thorough), max vision pages, and text batch size. Use augment mode to incrementally grow graphs instead of full rebuilds.

Known Risks

1) Incomplete model JSON
2) Long answer truncation
3) Vision over-processing
4) Context mismatch across chunks
5) UI state resets during reruns

Mitigations in App

Robust JSON extraction + fallbacks, answer continuation pass, page-level routing, citation enforcement, judge-and-retry loop, and stateful workspace selection.

Future Enhancements

1) Background worker queue for non-blocking chat/build. 2) Citation-level confidence scoring. 3) Table-optimized OCR pass. 4) Graph version snapshots and rollback. 5) Async streaming by token with smoother UI transitions.

Local-only Multi-model Persistent Graphs Grounded Chat