Project Title: InsightDoc AI: Multimodal Technical Assistant

Repository Name: insight-doc-ai

Tagline: A containerized, Zero-Cost RAG system for querying technical manuals with visual context and responsive design.

1. □ Project Overview

InsightDoc AI is an intelligent document assistant designed to solve the "Blind RAG" problem. Most AI tools only read text, ignoring the critical diagrams, charts, and schematics found in technical manuals.

This system uses a Multimodal Retrieval-Augmented Generation (RAG) architecture to process both text and images from PDFs and Word documents. It allows users to ask questions like *"What does the specific wiring diagram on page 5 show?"* and receive accurate answers with clickable citations that instantly navigate to the exact page (Smart Scrolling).

Data Strategy: To ensure replicability and copyright compliance, the system will be tested using a curated dataset of Open-Source Hardware Manuals (e.g., Fixit guides, Dell Service Manuals, and NASA Public Engineering Standards). These documents were selected specifically for their high density of visual data (schematics, wiring diagrams, and exploded views) to stress-test the multimodal capabilities.

2. 🏗 System Architecture & UI Design

This project demonstrates Full-Stack AI Engineering by combining a powerful backend with a professional, mobile-responsive frontend.

Backend (The Engine)

- **Framework: FastAPI (Python 3.10) for high-performance Async API.**
- **AI Engine: LangChain + Google Gemini 1.5 Flash (Multimodal, Fast, Free).**
- **Retrieval Pipeline: ChromaDB (Vector Search) + FlashRank (Reranker) for high-precision context filtering.**

- File Processing: Custom logic to convert .docx to .pdf instantly upon upload so "Smart Scrolling" works on all file types.
- Observability: Langfuse (Open Source) for tracing and cost monitoring.

Frontend (The Interface)

- Technology: Custom HTML5, CSS3, & Vanilla JavaScript.
- Desktop Layout (Split-Screen):
  - Left Sidebar (30%): Dedicated "Drag & Drop" zone for uploads (PDF/Docx) + File Status list.
  - Right Workspace (70%): Split view between the Chat Interface and the Smart PDF Viewer.
- Mobile Layout (Responsive):
  - Hamburger Menu: The Sidebar hides behind a slide-out drawer button (☰) to save space.
  - Tabbed Workspace: The view automatically stacks. Users see the Chat by default. When a citation [Page 5] is clicked, the specific PDF page slides into view, replacing the chat.

3. Key Engineering Features

- 2-Stage Retrieval (Reranking): Implements a "Search & Refine" architecture. First, ChromaDB retrieves broad matches. Then, FlashRank (a lightweight local Cross-Encoder) rescres and reorders results to ensure the most relevant context hits the LLM, significantly reducing hallucinations.
- Multimodal RAG Pipeline: Extracts and embeds both text and image descriptions using Gemini, allowing the AI to "see" charts and retrieve visual data.
- LLM Observability (Langfuse): uses the industry-standard Langfuse platform to trace query latency, token usage, and quality scores in real-time.

- Input Sanitization Pipeline: Implements a multi-layer security check (MIME-Type sniffing via python-magic + conversion firewall) to detect and block malicious uploads (e.g., renamed .exe files).
- Automated AI Quality Assurance (EvalOps): A standalone simulation module where a "User Agent" generates synthetic queries and a "Judge Agent" compares system outputs against a human-verified "Golden Dataset" (Answer Key).
- Streaming Response: The answer appears token-by-token (real-time typing) using Python Generators.
- Smart Page Scrolling: Citations (e.g., [Page 5]) are clickable links that instantly jump the PDF viewer to the specific source page.
- Docx-to-PDF Pipeline: Automatically converts Word documents to PDF on the server side to ensure consistent rendering.

4. 🛡 Risks & Hallucination Defense

- Data Poisoning Defense: Validates file binary headers (Magic Numbers) to prevent "File Masquerading" attacks.
- Reranking Filter: The FlashRank stage aggressively discards irrelevant documents before they reach the LLM context window.
- Prompt Constraints: System instructions strictly forbid guessing. If the context is missing, the AI returns "Data Not Found."
- Citation Enforcement: The model is programmatically forced to append [Page X] citations.

5. ▯ Directory Structure

Plaintext

```
insight-doc-ai/
├── Dockerfile          # Deployment instructions
├── README.md                 # The "Front Page" (Project manual & Setup)
├── AI_USAGE.md                 # AI Disclosure (Model details, Limitations, Ethics)
├── requirements.txt            # Dependencies (Added: flashrank, langfuse, python-magic)
├── main.py                # App Entry Point (FastAPI)
├── .env                   # API Keys (Git-ignored)
├── backend/
│   ├── rag_engine.py           # The AI Logic (Gemini + Chroma + FlashRank + Langfuse)
│   ├── file_processor.py       # Logic: PDF parsing, Docx Conversion, & Security Check
│   └── models.py          # Data Schemas
├── simulation/            # Quality Assurance Module
│   ├── sim_agent.py         # The "Virtual User" (Generates questions)
│   ├── evaluator.py        # The "Judge" (Grades using Reference-Based logic)
│   └── gold_standard.json    # Human-verified Answer Key for calibration
├── static/
│   ├── index.html         # Chat Interface (Mobile Responsive)
│   ├── style.css          # Styling (Media Queries for Mobile)
│   └── script.js          # Frontend Logic
└── data/                  # Storage for PDFs & DB
```

6. Implementation Roadmap

- **Phase 1: Setup. Initialize Git repo, Virtual Environment, and Folder Structure. (Status: Ready)**

- **Phase 2: The Brain. Build rag_engine.py (implementing FlashRank & Langfuse) and file_processor.py (Docx support & Security Layers).**
- **Phase 3: The API. Build main.py with FastAPI to serve the AI logic to the web.**
- **Phase 4: The Face. Build the HTML/JS frontend with CSS Media Queries and the Drag-and-Drop upload zone.**
- **Phase 5: The Analytics & EvalOps. Configure Langfuse dashboard and build the User Simulator to stress-test the system against the Golden Dataset.**
- **Phase 6: Deployment. Write the Dockerfile and push to Hugging Face.**