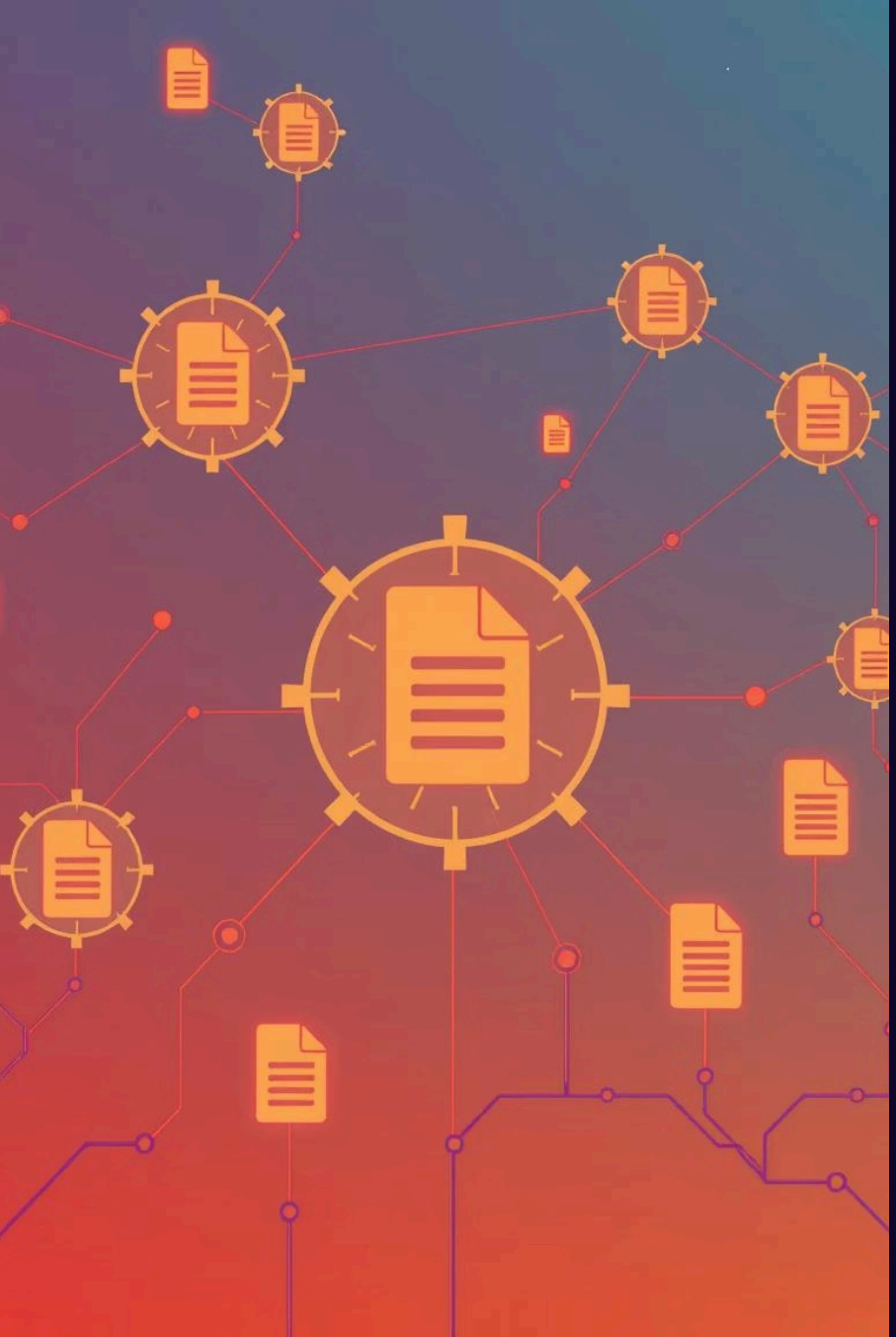


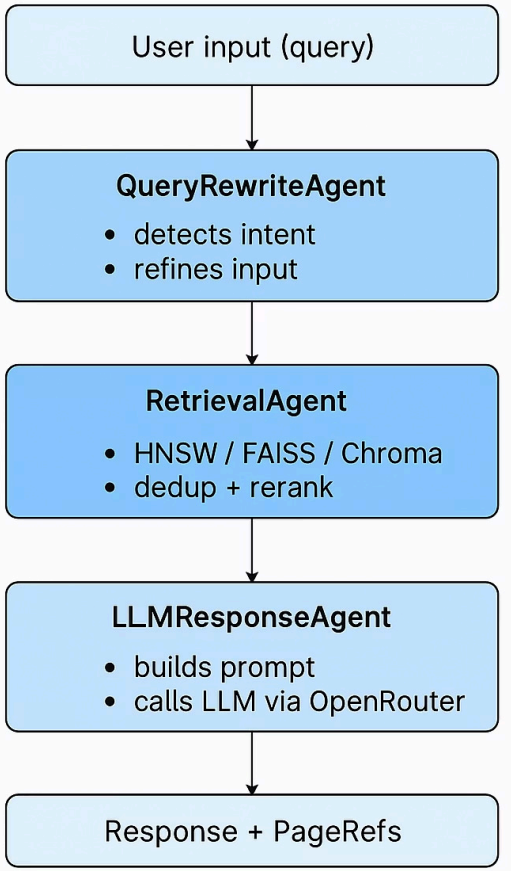
🧠 Docu-Mentor: Agentic RAG Chatbot for PDF QA

A modular, agent-based PDF Q&A system that uses semantic search, multi-vector retrieval, and LLM-driven response generation. Powered by OpenRouter, BGE/E5 embeddings, HNSW/FAISS/Chroma, and Streamlit UI.

📁 Project Directory Structure

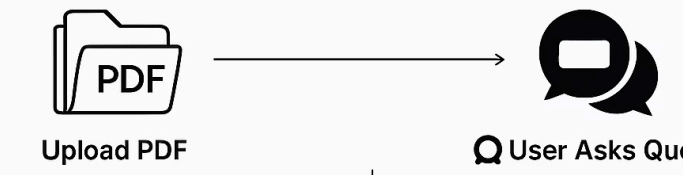
```
docu-mentor/
├── app.py # Streamlit app main entrypoint
├── chat.py # PDF viewer + chat UI
├── config.py # OpenRouter and model config
├── session_manager.py # (Optional) legacy session handling
├── viewer_component.py # PDF preview renderer
├── upload_modal.py # Upload widget (not always used)
├── requirements.txt # Dependencies
├── agents/
│   ├── ingestion_agent.py # Handles chunking and parsing
│   ├── embedding_agent.py # Embeds, deduplicates and stores
│   ├── query_rewrite_agent.py # Rewrites queries via LLM
│   ├── retrieval_agent.py # Retrieves chunks from vector stores
│   ├── llm_response_agent.py # Formats prompts and calls LLM
│   ├── reranker_agent.py # Re-ranks chunks using CrossEncoder or LLM
│   └── prompt_formatter_agent.py # (Optional) structured prompt formatting
├── core/
│   ├── agent_manager.py # Manages and shares all agents
│   ├── config_manager.py # Paths, constants, DB paths
│   ├── document_loader.py # Loads, parses, and chunks files
│   ├── embeddings.py # BGE/E5 embedding models and FAISS
│   ├── hnswlib_search.py # Fast HNSW searcher
│   ├── utils.py # Shared helpers (safe_execute, etc.)
│   └── mcp.py # Message creation helpers
├── utils/
│   └── page_utils.py # Page extraction and highlighting
├── models/ # Local models cache for transformers
├── vector_store/ # FAISS / HNSW / Chroma DBs
├── data/ # Uploaded PDF files
└── session_store.pkl # Optional chat memory
```

Agent based architecture with MCP integration



System Flow Diagram

System Flow Diagram



Upload PDF

- Stored into ***data***/
- Parsed with **`pdfplumber`**

User Asks Question

Sent through Query RewriteAgent to improve it

IngestionAgent

Splits into chünks
Adds source info (filename.pdf p. 3)

EmbeddingAgent

Embeds using *bge-base / e5-base*
Deduplicates
cosine similarity

RetrievalAgent

HNSW → FAISS → Chroma (falback)
Reranks with ColBERT, then LLM or CrossEncoder

LLMResponse-Agent

Builds context-rich prompt
Calls GPT-4 / Claude via OpenRouter

Response Rendered

In chat bubble
Below it: *References*

⚙️ Core Technologies Powering Docu-Mentor

LLMs & API Integration

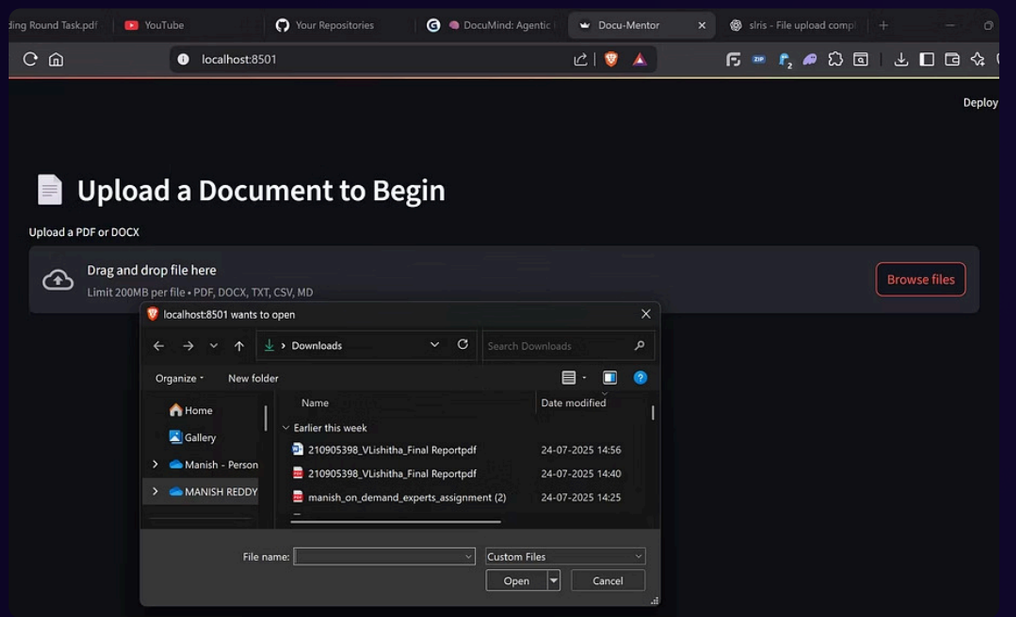
Utilizing OpenRouter, DocuMind accesses diverse Large Language Models for rapid and cost-efficient AI-driven insights, ensuring optimal performance for document queries.

Vector Databases & Indexing

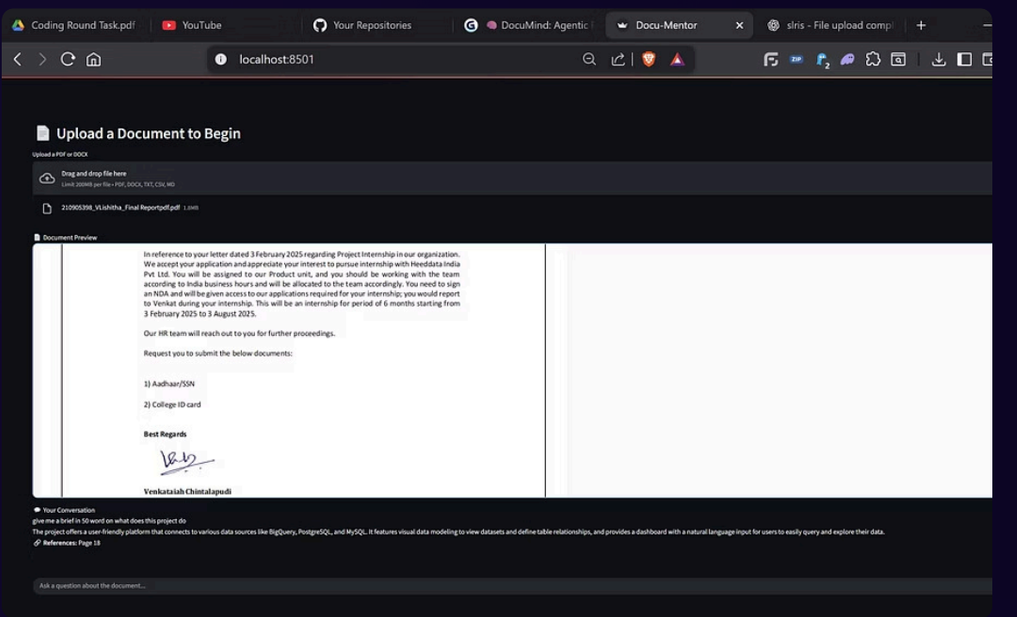
ChromaDB provides persistent storage, while FAISS and HNSWLib enable high-performance approximate nearest neighbor search for swift and precise retrieval of document embeddings.

Embedding Models

DocuMind employs BGE and E5 models to transform text into high-quality numerical embeddings, capturing precise semantic meaning for effective search and retrieval.



We can upload files by clicking the upload button and selecting them



Streamlit powers the interactive web UI, while CrossEncoder Models refine retrieval relevance, ensuring users receive the most pertinent information from their queries.

🔄 How It Works (From Upload to Answer)

1

📂 Upload PDF

Stored in `data/`, parsed into text using pdfplumber.

2

📄 IngestionAgent

Splits into chunks, adds source info (filename.pdf p. 3).

3

🔍 EmbeddingAgent

Embeds using bge-base / e5-base, deduplicates, saves in FAISS, Chroma, HNSW.

4

💬 User Asks Question

Sent through QueryRewriteAgent to improve it.

5

📡 RetrievalAgent

HNSW → FAISS → Chroma (fallback), reranks with ColBERT, then LLM or CrossEncoder.

6

🧠 LLMResponseAgent

Builds context-rich prompt, calls GPT-4 / Claude via OpenRouter.

7

🗣️ Response Rendered

In chat bubble, with clickable reference pages below.

🚧 Challenges Faced

Agent Communication

Managing typed messages across various agents (MCP-style architecture).

Vector Store Orchestration

Handling different vector stores with robust fallback mechanisms for retrieval.

Smart Deduplication

Building chunk deduplication that respects semantic similarity during ingestion.

Reference Mapping

Rendering accurate page-level references with proper mapping from chunks.

File Handling

Addressing complex file handling issues with Streamlit and browser PDF viewers.

🌱 Future Improvements

Orchestration

Integrate LangGraph or Haystack for advanced agent orchestration.

User Feedback

Add user feedback grading and automatic response regeneration.

Multi-Turn Memory

Implement cross-file multi-turn conversational memory for continuity.

OCR Support

Enhance the system with image OCR for scanned PDF documents.