DOCINFERX : LOCAL AI-POWERED DOCUMENT INTELLIGENCE SYSTEM

STUDENT: SHASHANK SHEKHAR CHOUDHARY

REG. NO.: 25BCY10189

COURSE: FUNDAMENTALS IN AI AND ML

FACULTY: VIVEK

INSTITUTE: VIT BHOPAL UNIVERSITY

SEMESTER: 1ST SEM

SUBMISSION DATE: 24.11.2025

INTRODUCTION

- Modern documents contain a large amount of unstructured information spread across PDFs, images, scanned pages. Extracting meaning from these documents manually is slow and inefficient.

- DocInferX is a local RAG-based system that allows users to upload documents, process them using OCR and embeddings, and then ask natural-language questions to retrieve precise answers entirely offline on local machine.

- The project demonstrates real-world application of NLP, vector databases, and information retrieval.

PROBLEM STATEMENT

Individuals and professionals struggle to extract useful information from large multi-page PDFs, scanned documents, and images. Searching manually is slow, error-prone, and computationally expensive. There is a need for a local, private, offline system that can:

- Ingest PDFs and images
- Extract and clean text
- Convert content into embeddings for fast retrieval
- Answer user questions accurately
- Provide a modern, intuitive interface

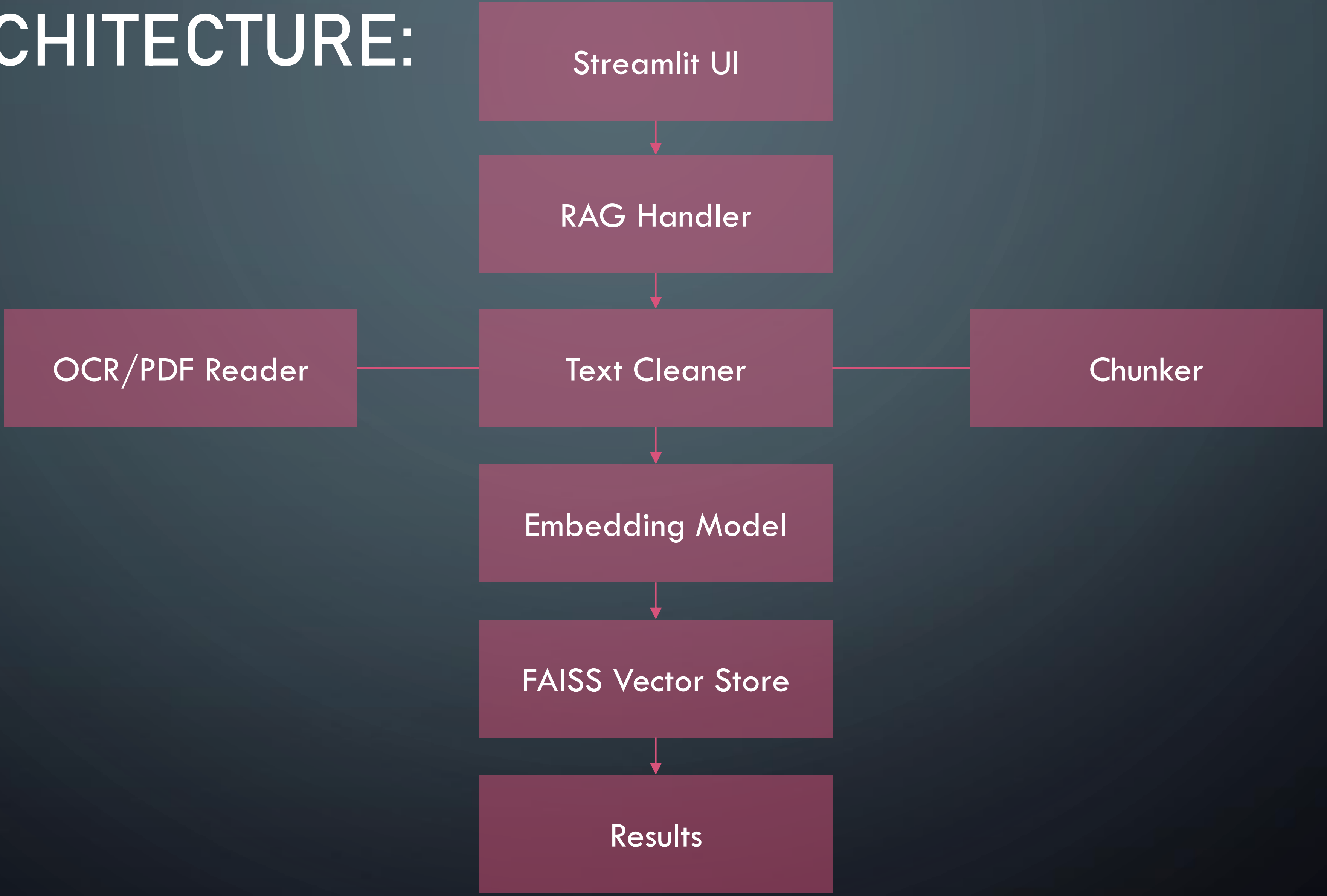
FUNCTIONAL REQUIREMENTS

1. Document Ingestion Module
   - Accepts PDF or image files
   - Extracts text via PDF parser or OCR
   - Cleans and preprocesses content
2. Embedding & Indexing Module
   - Splits text into chunks
   - Generates vector embeddings using Phi-2
   - Stores them in FAISS vector store
3. RAG Query Module
   - Accepts natural-language queries
   - Retrieves relevant chunks
   - Generates response using LLM
   - Displays clean, readable answers

NON-FUNCTIONAL REQUIREMENTS

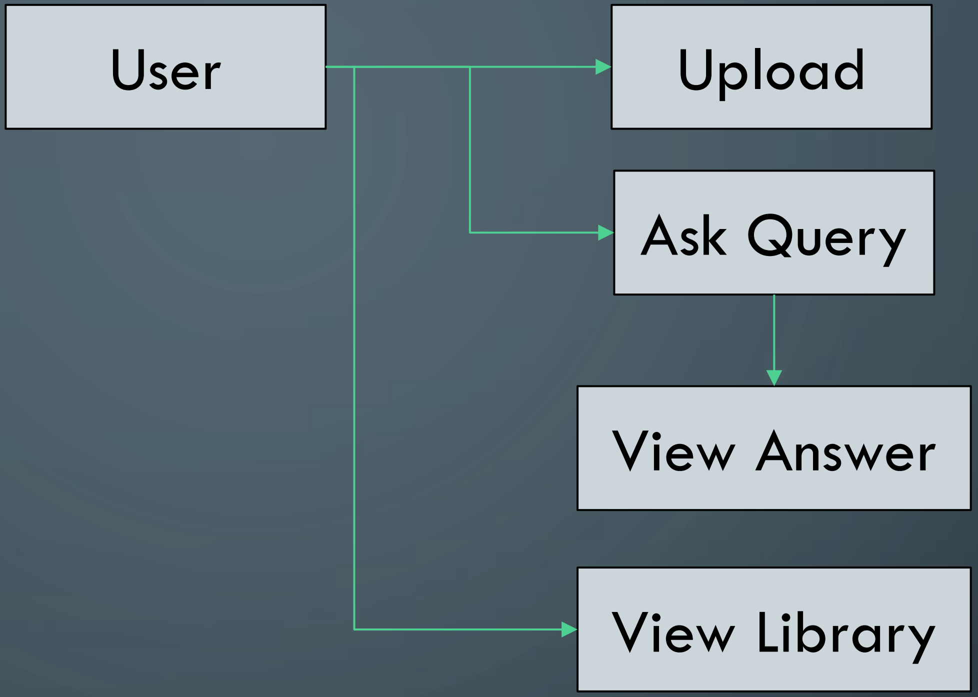
1. **Usability** – Clean Streamlit UI with modern design.
2. **Performance** – FAISS vector search ensures fast retrieval even for large documents.
3. **Maintainability** – Modular folder structure with separate components.
4. **Reliability** – Handles corrupted PDFs, missing pages, and OCR failures.
5. **Resource Efficiency** – Runs fully offline with GPU requirement.
6. **Security** – No cloud upload; all processing is local.

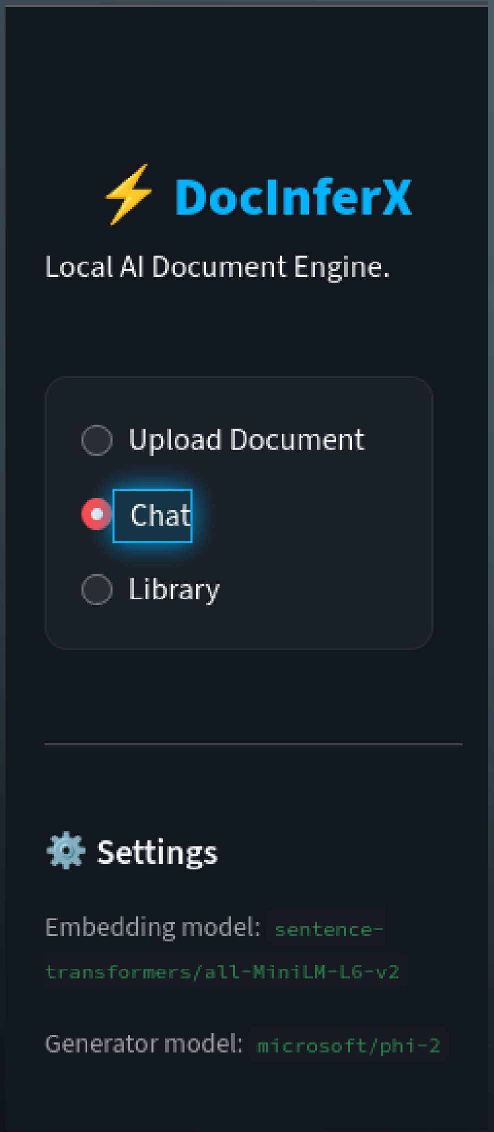
SYSTEM ARCHITECTURE:



Design Diagrams

- Use Case Diagram





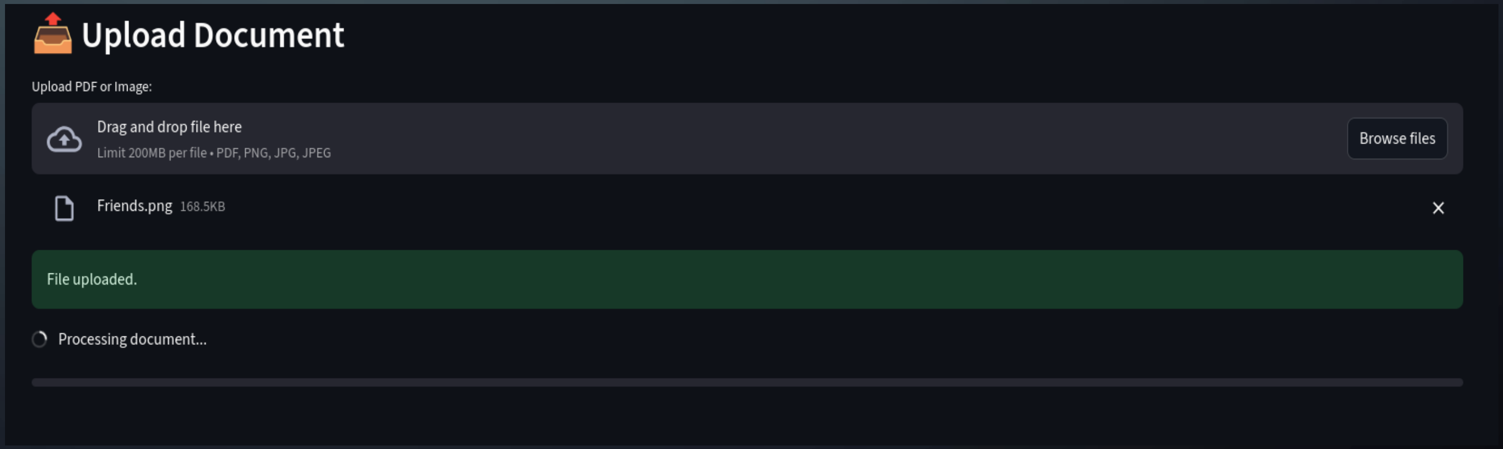
Upload Document

[✓ Chat]

Library

Embedding model: `sentence-transformers/all-MiniLM-L6-v2`

Generator model: `microsoft/phi-2`



Upload PDF or Image:

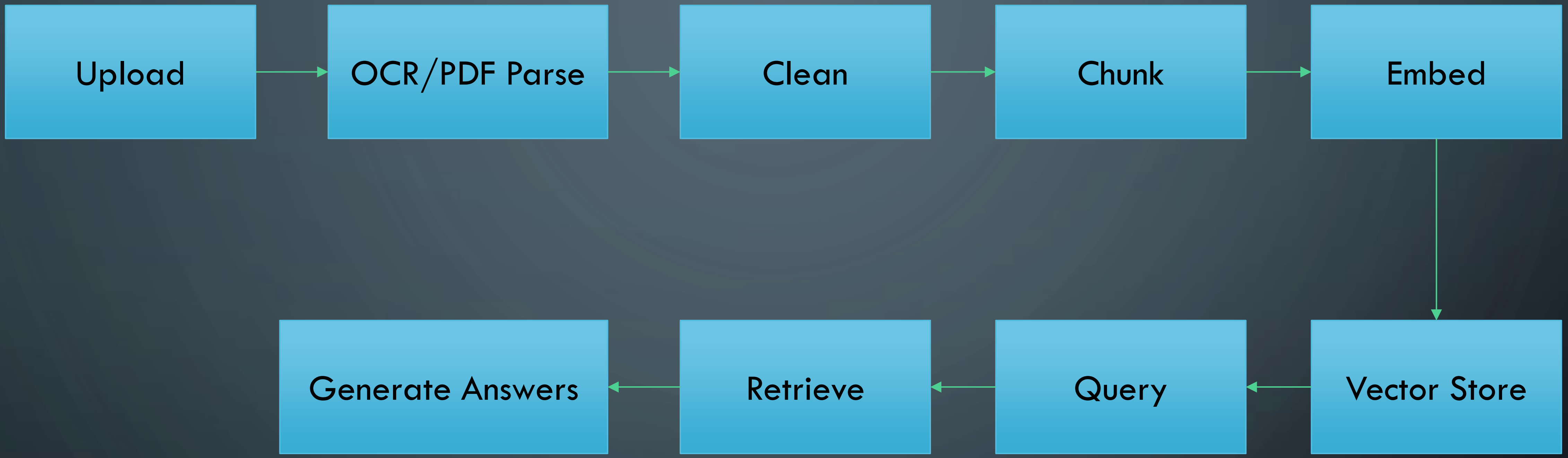
Drag and drop file here
Limit 200MB per file • PDF, PNG, JPG, JPEG

Friends.png 168.5KB

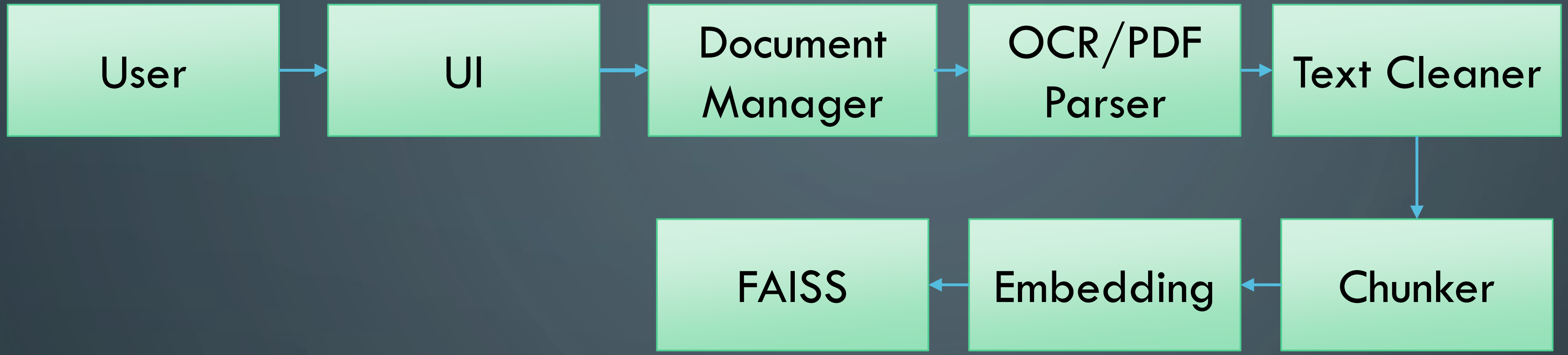
File uploaded.

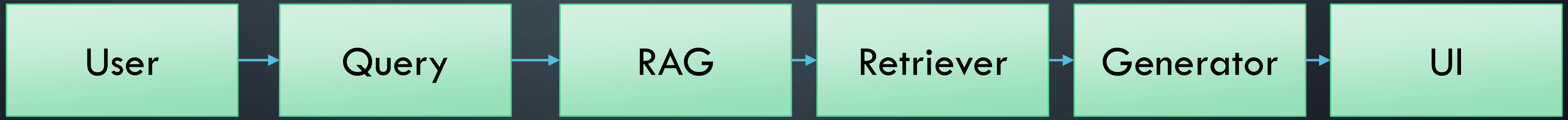
Processing document...

- Workflow Diagram



- Sequence Diagram

1.


2.


• Class/Component Diagrams

```
DocInferX/
│
├──── streamlit_app.py      # Main UI
├──── config.py             # Global config
├──── cli_test.py           # CLI test script
│
├──── embed/
│     ├──── vectorizer.py       # Embeddings
│     └──── chunker.py          # Text chunking
│
├──── preprocess/
│     ├──── pdf_reader.py       # PDF parsing
│     ├──── ocr.py              # OCR engine
│     └──── cleaner.py          # Text cleanup
│
├──── library/
│     ├──── manager.py          # Document ingestion
│     └──── metadata.py         # Meta store
│
├──── rag/
│     └──── pipeline.py         # RAG inference
│
├──── Samples/              # Example files
│
├──── data/                 # Vector store + metadata
│
├──── requirements.txt
└──── Dockerfile
```

DESIGN DECISIONS & RATIONALE

1. **Local-first architecture:** Ensures privacy and offline access.

2. **FAISS:** Fast similarity search for large text corpora.

3. **Phi-2:** Lightweight and performant LLM suitable for on-device generation.

4. **Streamlit:** Simplifies UI without needing frontend frameworks.

5. **Modular folder structure:** Easier debugging and future scalability.

Implementation Details

- **Python backend with separate packages:**

  /preprocess (OCR, cleaning, PDF reading)

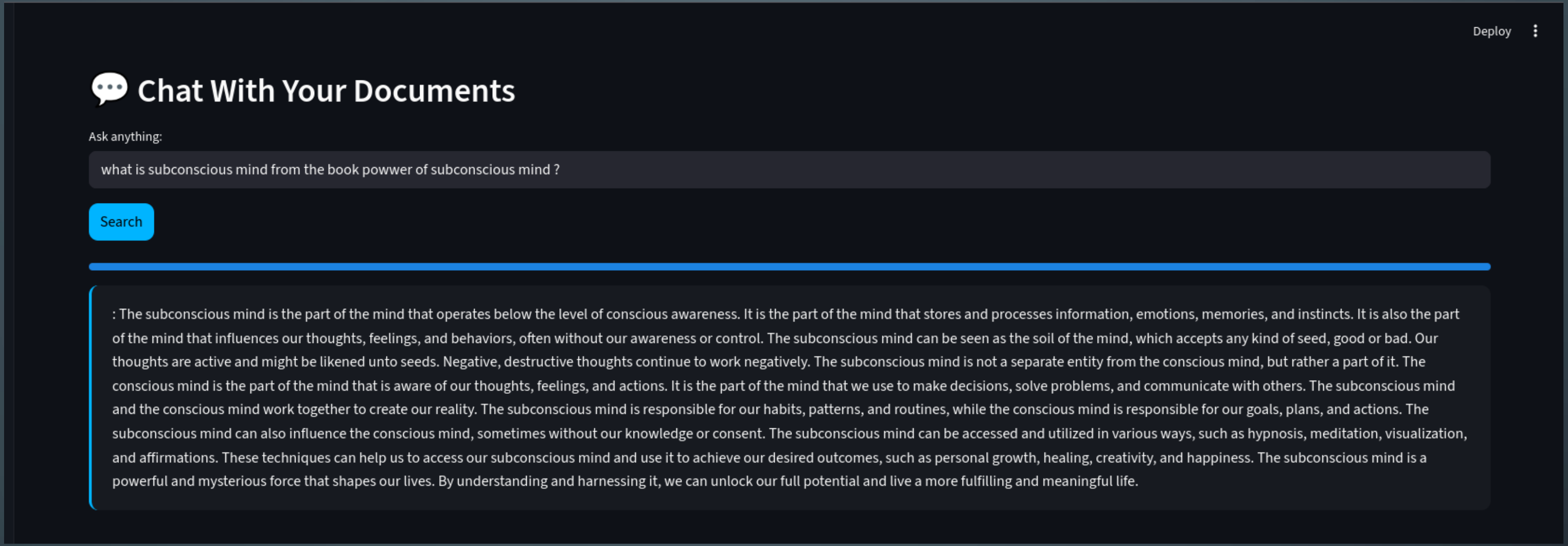
  /embed (chunker, vectorizer)

  /rag (retrieval pipeline)

  /library (metadata and document management)

- **UI built in Streamlit**
- **Vector DB: FAISS**
- **OCR: PaddleOCR**
- **Embeddings: Phi-2**
- **Storage: Local JSON metadata + FAISS index**

Screenshots


Deploy

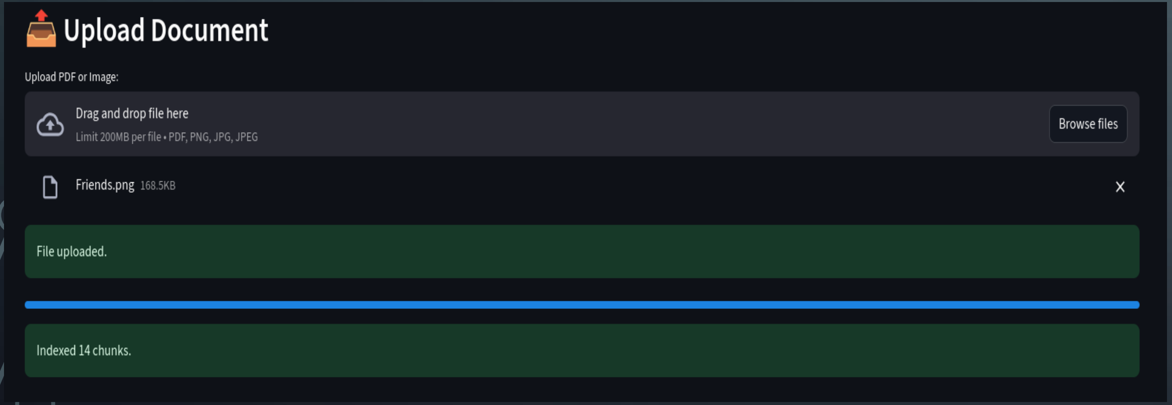
💬 Chat With Your Documents

Ask anything:

what is subconscious mind from the book powwer of subconscious mind ?

Search

: The subconscious mind is the part of the mind that operates below the level of conscious awareness. It is the part of the mind that stores and processes information, emotions, memories, and instincts. It is also the part of the mind that influences our thoughts, feelings, and behaviors, often without our awareness or control. The subconscious mind can be seen as the soil of the mind, which accepts any kind of seed, good or bad. Our thoughts are active and might be likened unto seeds. Negative, destructive thoughts continue to work negatively. The subconscious mind is not a separate entity from the conscious mind, but rather a part of it. The conscious mind is the part of the mind that is aware of our thoughts, feelings, and actions. It is the part of the mind that we use to make decisions, solve problems, and communicate with others. The subconscious mind and the conscious mind work together to create our reality. The subconscious mind is responsible for our habits, patterns, and routines, while the conscious mind is responsible for our goals, plans, and actions. The subconscious mind can also influence the conscious mind, sometimes without our knowledge or consent. The subconscious mind can be accessed and utilized in various ways, such as hypnosis, meditation, visualization, and affirmations. These techniques can help us to access our subconscious mind and use it to achieve our desired outcomes, such as personal growth, healing, creativity, and happiness. The subconscious mind is a powerful and mysterious force that shapes our lives. By understanding and harnessing it, we can unlock our full potential and live a more fulfilling and meaningful life.


📤 Upload Document

Upload PDF or Image:

Drag and drop file here
Limit 200MB per file • PDF, PNG, JPG, JPEG

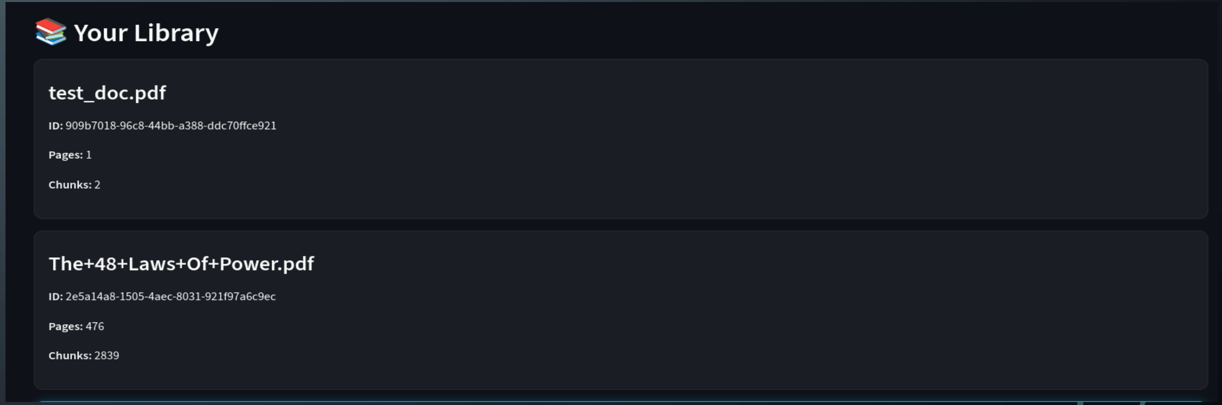
Browse files

Friends.png 168.5KB

X

File uploaded.

Indexed 14 chunks.


📚 Your Library

test_doc.pdf

ID: 909b7018-96c8-44bb-a388-ddc70ffce921

Pages: 1

Chunks: 2

The+48+Laws+Of+Power.pdf

ID: 2e5a14a8-1505-4aec-8031-921f97a6c9ec

Pages: 476

Chunks: 2839

Testing Approach

- Tested with multiple PDFs: books, handwritten scans, power-law dense PDFs.
- Tested OCR accuracy on blurred images.
- Verified embedding and chunk alignment.
- Checked retrieval accuracy with different queries.
- Checked UI on Linux + Windows.
- Docker test for portability.

Challenges Faced

- Handling large PDFs with mixed text + images
- Chunking strategy for semantic coherence
- Optimizing OCR speed
- Avoiding cache corruption
- Designing a modern UI in Streamlit
- Mismatch of python modules versions
- VRAM management to make it Local on device

Learnings & Key Takeaways

- Practical understanding of RAG systems
- Using FAISS and embeddings
- Implementing OCR pipelines
- Multi-module project structuring
- Version control workflows (GitHub)
- Dockizing Python applications
- UI/UX considerations even in backend-heavy projects

FUTURE ENHANCEMENTS

- Support for audio transcription
- Summarization mode
- Cloud-sync optional mode
- Fine-tuned local model for document answering
- Increasing response speed
- Reducing hallucinations

References

- Faiss documentation
- HuggingFace Transformers
- PaddleOCR documentation
- Streamlit documentation
- Various research papers on RAG