MINI PROJECT REPORT

ON

“AI PDF Reader”

SUBMITTED IN

COMPUTER SCIENCE & ENGINEERING

BACHELOR OF TECHNOLOGY

BY

AMAN KANNAUJIYA

(ROLL NO: [22150010004])

SEMESTER VI

SUBMITTED

TO

MR. GAURAV DWIVEDI

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

UNITED UNIVERSITY,

RAWAPUR, PRAYAGRAJ, UTTAR PRADESH-211012

ABSTRACT

This project presents the development of an AI-powered PDF Reader designed to enhance document interaction through natural language processing. Unlike traditional PDF viewers that require manual navigation and keyword search, this AI-based system enables users to query and understand content conversationally. By leveraging machine learning models such as transformer-based language models (e.g., GPT or BERT), the system extracts, indexes, and interprets PDF content efficiently. The application supports semantic understanding, context-aware answers, and can summarize large documents or highlight relevant sections based on user queries. This innovation significantly improves user experience, especially in research, legal, academic, and corporate environments where large documents are prevalent. The project demonstrates how artificial intelligence can streamline information retrieval and make document analysis more intuitive and accessible.

INTRODUCTION:

PDF files are commonly used for sharing documents, but extracting specific information from them can be difficult and time-consuming. Traditional PDF readers require manual searching and lack intelligent features. This project introduces an AI-powered PDF Reader that uses natural language processing (NLP) to allow users to ask questions and receive meaningful answers from the document. It makes reading and analyzing PDFs faster, easier, and more interactive by understanding the context and content of the text.

MOTIVATION:

Reading and searching through long PDF documents can be slow and frustrating. Traditional PDF readers do not understand the content, making it hard to find specific information quickly. This project is motivated by the need to make PDF interaction smarter and faster using AI, allowing users to ask questions and get instant, accurate answers from the document.

WHY THIS PROJECT?

This project was chosen to solve a common problem faced by students, researchers, and professionals—difficulty in quickly finding relevant information in large PDF documents. By using AI and natural language processing, the project offers a smarter way to interact with PDFs, making reading, searching, and understanding content much easier. It combines practical usefulness with modern technology, making it both innovative and impactful.

OBJECTIVES AND GOALS:

- To develop an AI-powered PDF Reader that can understand and respond to user queries in natural language.
- To automate the process of extracting key information from large PDF documents.
- To enhance the user experience by enabling semantic search and intelligent navigation.
- To implement summarization and question-answering features for faster content understanding.
- To create a user-friendly interface that makes document interaction simple and efficient.

BACKGROUND:

PDFs are commonly used for sharing documents, but finding specific information in large files is often difficult with traditional readers. These tools lack understanding of the content and only support basic search functions. With the rise of AI and natural language processing (NLP), it's now possible to build systems that understand text and respond intelligently. This project uses these technologies to create a smarter PDF reader that makes document navigation and information retrieval easier.

Tools and Platform:

The project uses a modern full-stack architecture:

- **Frontend:** Next.js, React, TypeScript, Tailwind CSS for UI, and Lucide React for icons.
- **Backend:** Node.js, Express.js, Multer for file handling, BullMQ & Redis for job queuing.
- **AI & Document Processing:** LangChain, OpenAI GPT-4.1, and Qdrant for semantic search and AI-powered report generation.
- **API Endpoints:** /upload/pdf, /chat, and /report for document interaction.
- **Development Tools:** PNPM for package management, ESLint for code quality.

This stack ensures efficient document processing, semantic search, and AI integration.

Hardware requirements:

 CPU: A multi-core processor (4+ cores) is recommended for handling AI processing and asynchronous tasks efficiently.

 RAM: At least 8GB of RAM for smooth operation, especially when processing large PDF files and running AI models.

 Storage: SSD with at least 100GB of free space to store documents, database files, and models.

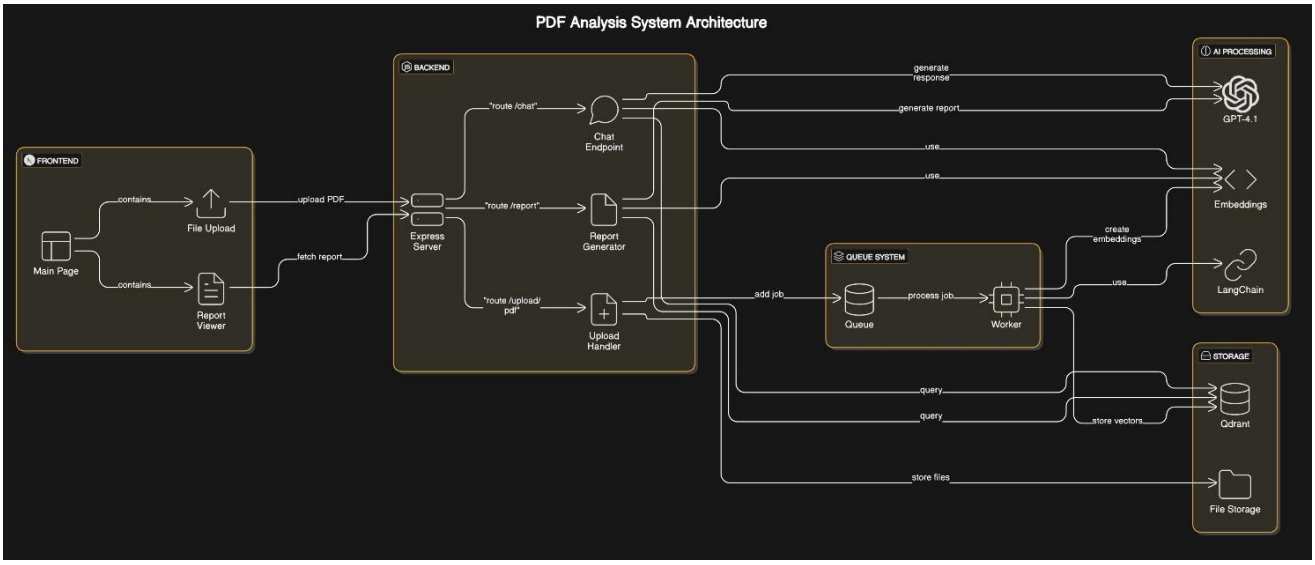
 Network: A stable internet connection is required for AI model inference (if hosted remotely) and data transfer.

Software platforms:

- Frontend : Next js , Tailwind-CSS, Lucide react
- Backend : Express , node , typescript
- Database server : VectorDB(Qdrant)
- AI_Engine : Langflow
- OpenAI

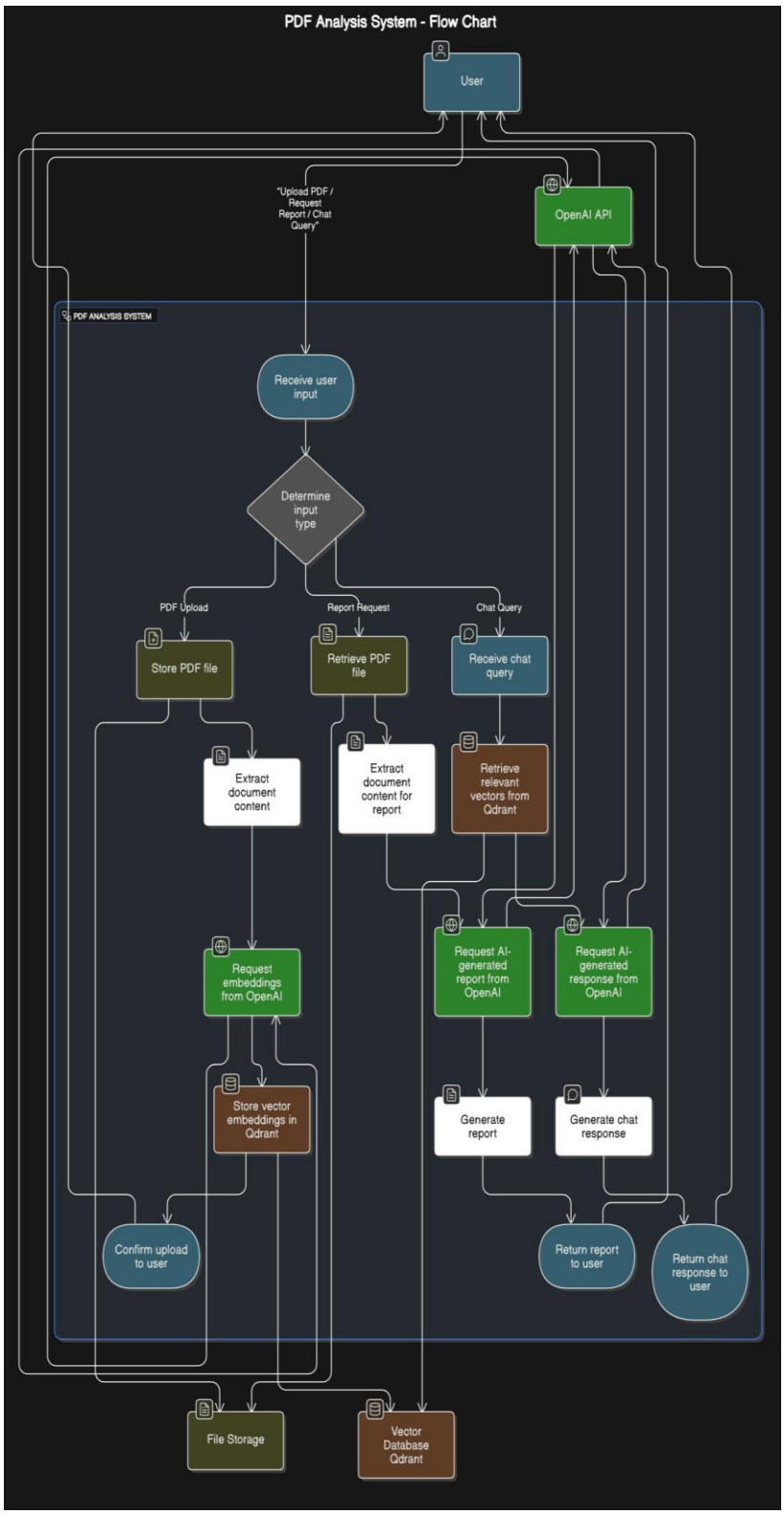
SRS DIAGRAMS:

- **UML DIAGRAM:**

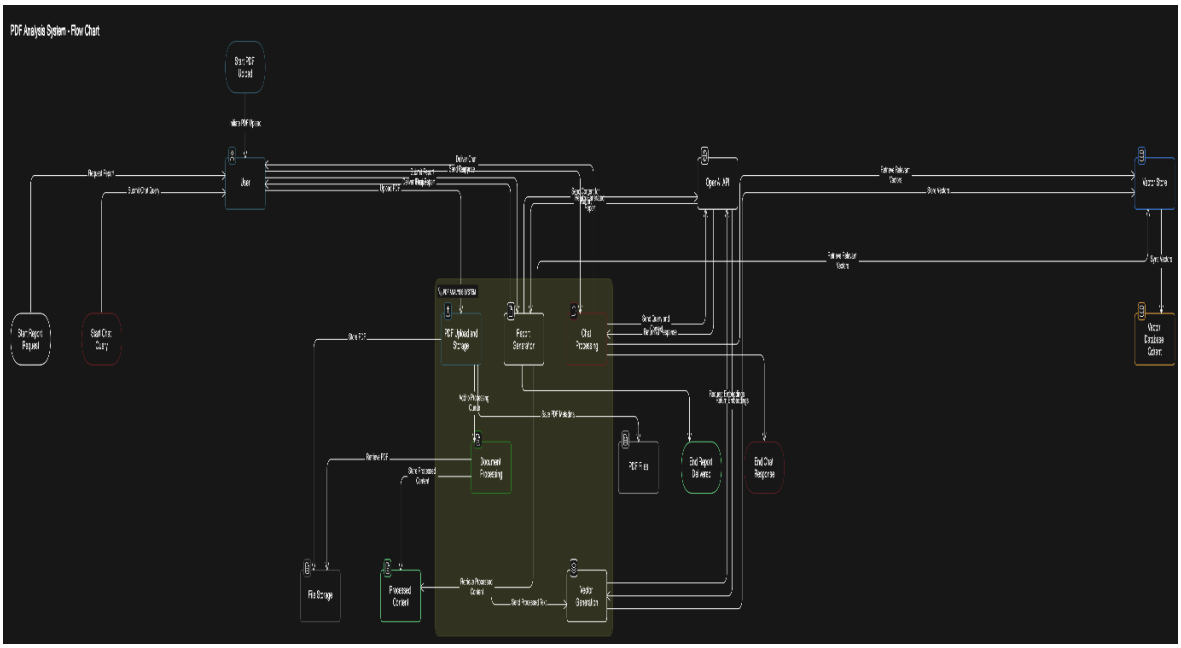


DFD:

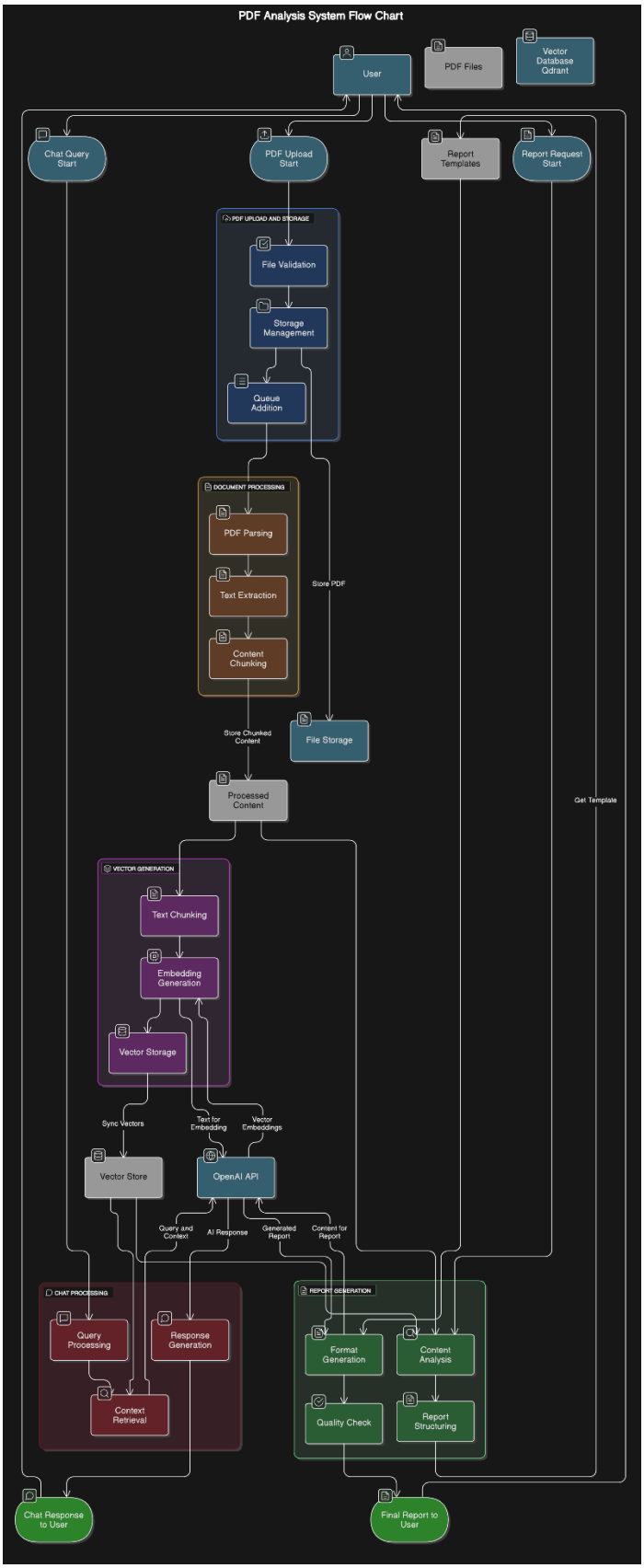
- 0 LEVEL


PDF Analysis System - Flow Chart
User
"Upload PDF / Request Report / Chat Query"
OpenAI API
PDF ANALYSIS SYSTEM
Receive user input
Determine input type
PDF Upload
Report Request
Chat Query
Store PDF file
Retrieve PDF file
Receive chat query
Extract document content
Extract document content for report
Retrieve relevant vectors from Qdrant
Request embeddings from OpenAI
Request AI-generated report from OpenAI
Request AI-generated response from OpenAI
Store vector embeddings in Qdrant
Generate report
Generate chat response
Confirm upload to user
Return report to user
Return chat response to user
File Storage
Vector Database Qdrant

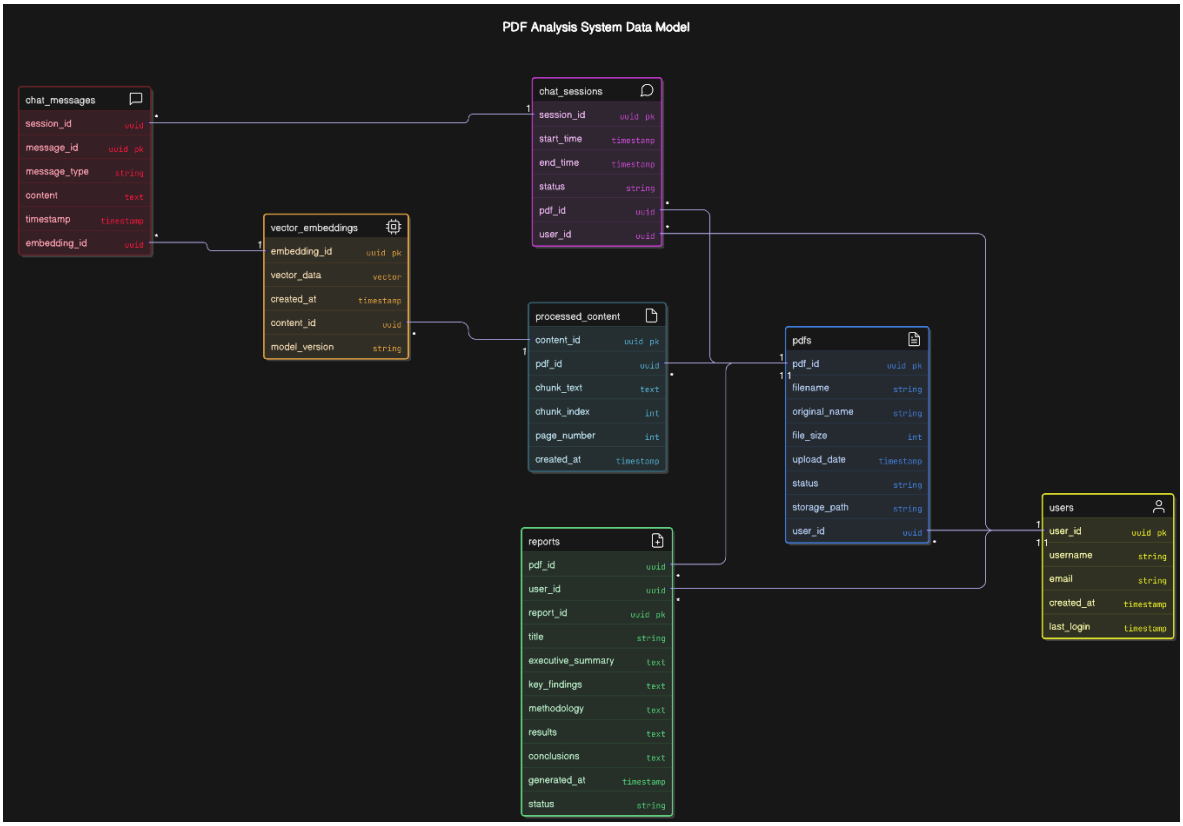
- 1 LEVEL


PDF Analysis System - Flow Chart
User
OpenAI API
Vector Store
PDF Upload and Storage
Report Generator
Chat Processing
Document Processing
PDF Files
End Report Delivery
End Chat Response
File Storage
Processed Content
Vector Embedding

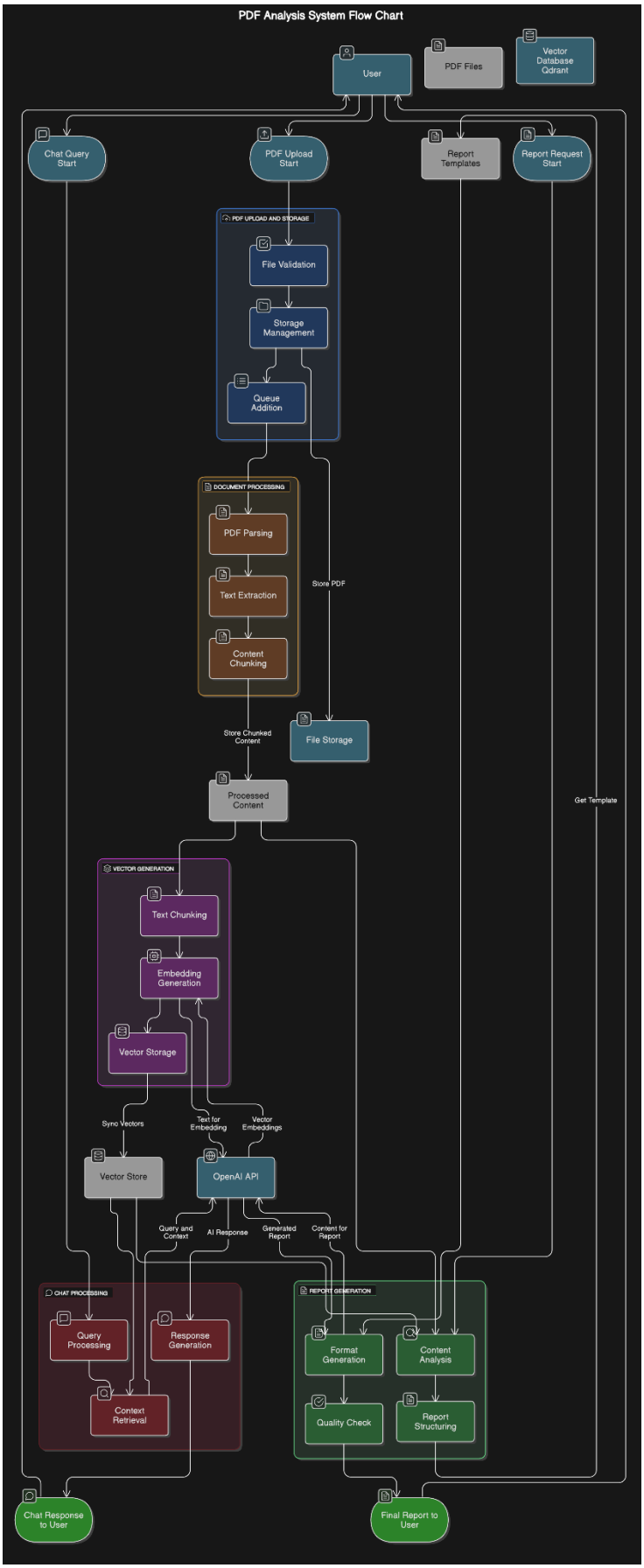
2 LEVEL


PDF Analysis System Flow Chart
User
PDF Files
Vector Database Qdrant
Chat Query Start
PDF Upload Start
Report Templates
Report Request Start
PDF UPLOAD AND STORAGE
File Validation
Storage Management
Queue Addition
DOCUMENT PROCESSING
PDF Parsing
Text Extraction
Content Chunking
Store PDF
Store Chunked Content
File Storage
Processed Content
Get Template
VECTOR GENERATION
Text Chunking
Embedding Generation
Vector Storage
Sync Vectors
Text for Embedding
Vector Embeddings
Vector Store
OpenAI API
Query and Context
AI Response
Generated Report
Content for Report
CHAT PROCESSING
Query Processing
Response Generation
Context Retrieval
REPORT GENERATION
Format Generation
Content Analysis
Quality Check
Report Structuring
Chat Response to User
Final Report to User

ER DIAGRAM

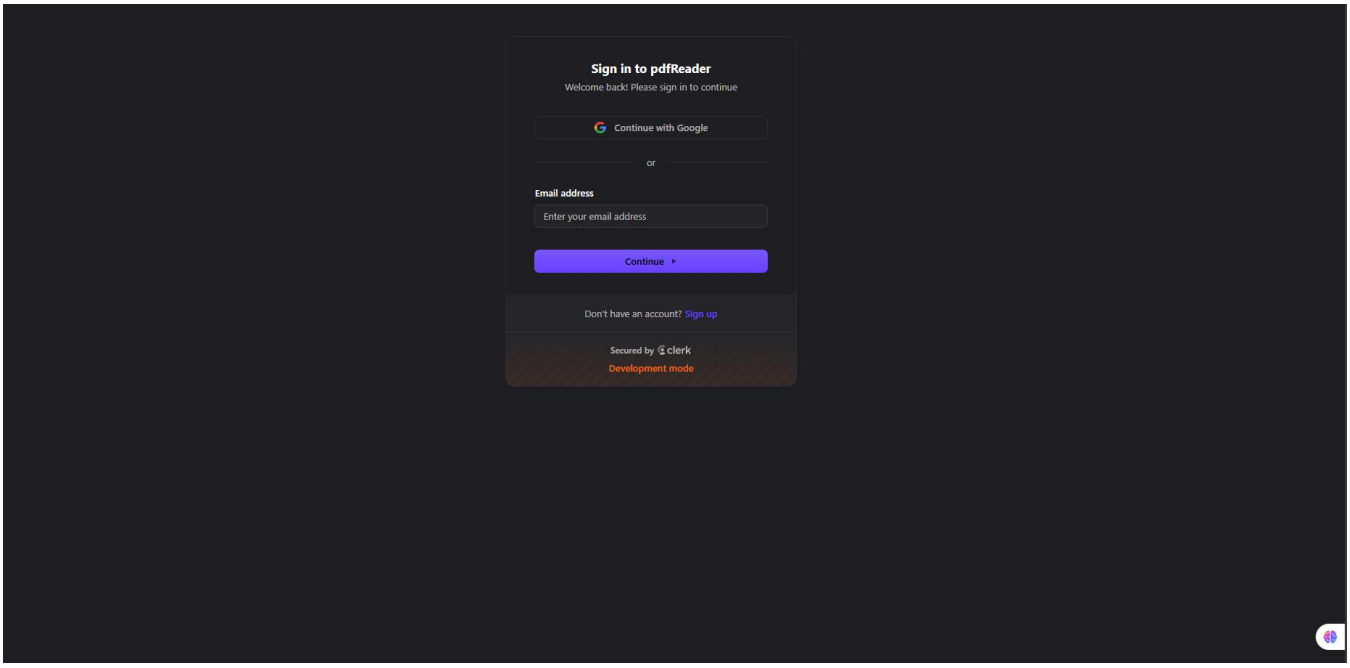

PDF Analysis System Data Model
chat_messages
session_id uuid
message_id uuid pk
message_type string
content text
timestamp timestamp
embedding_id uuid
chat_sessions
session_id uuid pk
start_time timestamp
end_time timestamp
status string
pdf_id uuid
user_id uuid
vector_embeddings
embedding_id uuid pk
vector_data vector
created_at timestamp
content_id uuid
model_version string
processed_content
content_id uuid pk
pdf_id uuid
chunk_text text
chunk_index int
page_number int
created_at timestamp
pdfs
pdf_id uuid pk
filename string
original_name string
file_size int
upload_date timestamp
status string
storage_path string
user_id uuid
users
user_id uuid pk
username string
email string
created_at timestamp
last_login timestamp
reports
pdf_id uuid
user_id uuid
report_id uuid pk
title string
executive_summary text
key_findings text
methodology text
results text
conclusions text
generated_at timestamp
status string

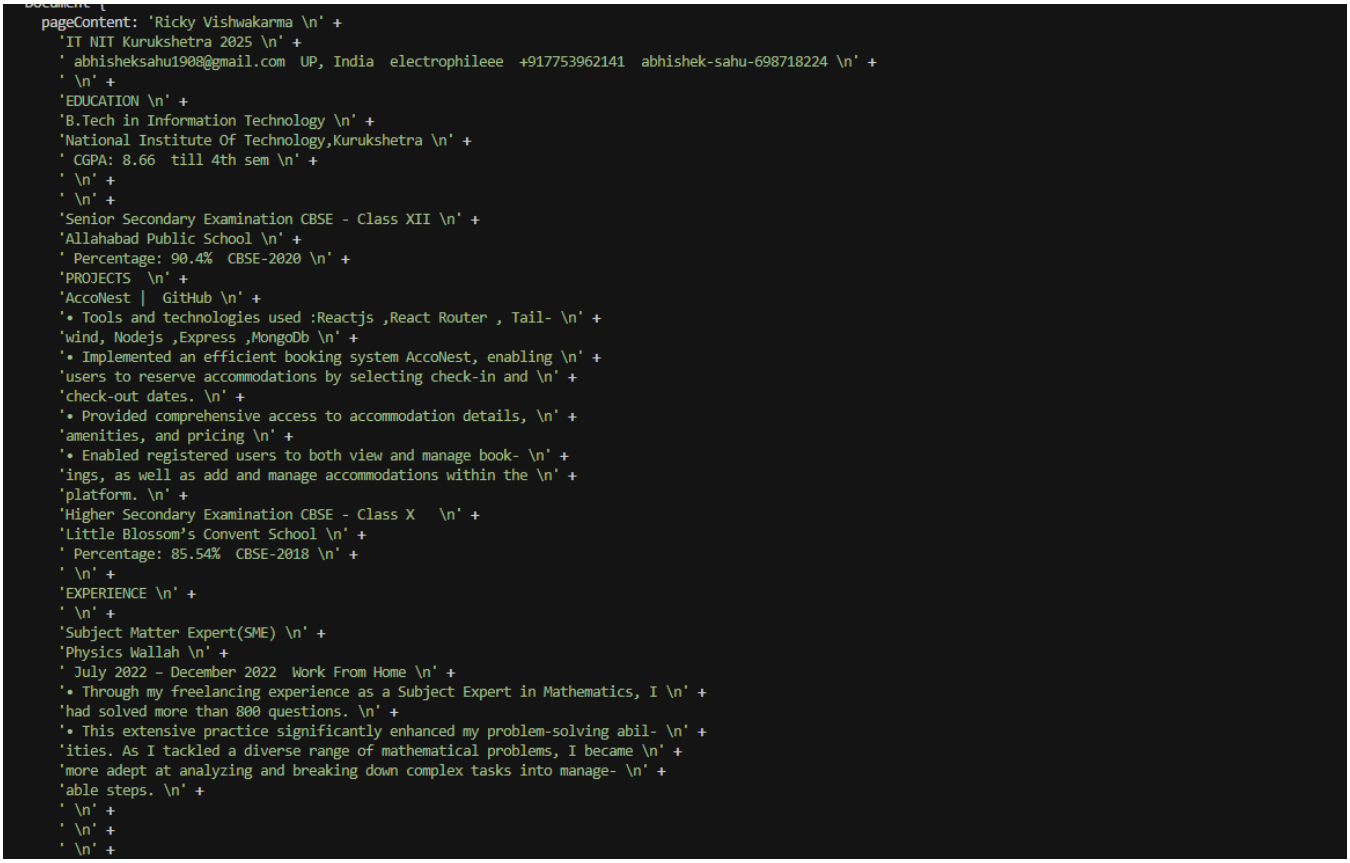
SCREENSHOT OF PROJECT



SCREENSHOT


Sign in
Sign up
File Upload
Section 2


Sign in to pdfReader
Welcome back! Please sign in to continue
Continue with Google
or
Email address
Enter your email address
Continue
Don't have an account? Sign up
Secured by clerk
Development mode



APPLICATIONS:

1. **PDF Document Processing**
2. **Semantic Search and analysis**
3. **AI-powered questioning**
4. **Asynchronous process**

Future of the project:

1. **Enhanced Document Processing**
2. **Improve AI capabilities**

REFERENCES:-

VectorDB Docs
Docker docs
Next js documentation
Mozilla dev networks
Redis docs