Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core components and implementation details for **PDFQuery**, a PDF-based Question Answering System. The system enables users to upload PDF documents and ask questions, with answers generated using Retrieval-Augmented Generation (RAG) via LangChain and OpenAI. The solution is built using Python and Flask.

---

1. System Components

Component	Description	Key Responsibilities
Web API (Flask)	RESTful API for user interaction	File upload, question submission, responses
PDF Processor	Handles PDF parsing and text extraction	Extracts and preprocesses PDF content
Vector Store	Stores and retrieves document embeddings	Indexing, similarity search
RAG Engine	Orchestrates retrieval and answer generation	Query vector store, call LLM, format answer
OpenAI Connector	Interface to OpenAI LLM APIs	Generate answers from context + question

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`PDFUploader`	Handles PDF file uploads	`save_file()` , `validate_file()`
`PDFParser`	Extracts text from PDFs	`extract_text(file_path)`
`VectorStore`	Manages embeddings and retrieval	`add_document(text)` , `query(query)`
`RAGEngine`	Main QA logic	`answer_question(question, doc_id)`
`OpenAIClient`	LLM API wrapper	`generate_answer(context, question)`

Relationships:

- `PDFUploader` → `PDFParser` → `VectorStore`
- `RAGEngine` uses `VectorStore` and `OpenAIClient`

---

3. Data Structure Overview

Model	Fields / Structure
Document	`id` , `filename` , `text` , `embedding`

Question	`id` , `user_id` , `document_id` , `question_text`
Answer	`id` , `question_id` , `answer_text` , `source_chunks`

---

4. Algorithms / Logic

Question Answering Flow:

```
def answer_question(question, document_id):
    context_chunks = VectorStore.query(question, document_id)
    answer = OpenAIClient.generate_answer(context_chunks, question)
    return answer
```

PDF Upload & Indexing:

```
def upload_pdf(file):
    path = PDFUploader.save_file(file)
    text = PDFParser.extract_text(path)
    VectorStore.add_document(text)
```

---

5. Error Handling

Scenario	Handling Approach
Invalid PDF upload	Return 400 error, log and notify user
PDF parsing failure	Return 422 error, log error
Embedding/Vector store failure	Retry, else return 500 error
OpenAI API failure	Retry with backoff, else return 503 error
Question with no context found	Return informative message to user

---

End of Document