High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components of **PDFQuery**, a PDF-based Question Answering System for the education sector. PDFQuery enables users to upload PDF documents and ask questions, with answers generated using Retrieval-Augmented Generation (RAG) techniques via LangChain and OpenAI.

---

1. System Architecture Overview

Architecture Description:
PDFQuery is a web-based application built with Python and Flask, integrating LangChain and OpenAI APIs for generative question answering. The system consists of a frontend interface, backend server, PDF processing pipeline, and external AI services.

Main System Components:

Component	Role
Web Frontend	User interface for PDF upload and question submission
Flask Backend API	Handles requests, orchestrates processing, and manages user sessions
PDF Processor	Extracts and preprocesses text from uploaded PDFs
RAG Engine (LangChain + OpenAI)	Retrieves relevant context and generates answers
Storage	Temporarily stores PDFs and extracted data

---

2. Component Interactions

Step	Interaction Description
1	User uploads PDF and submits a question via the Web Frontend
2	Flask Backend receives request, stores PDF, and invokes PDF Processor
3	PDF Processor extracts text and passes it to the RAG Engine
4	RAG Engine (LangChain + OpenAI) retrieves relevant context and generates an answer
5	Flask Backend returns the generated answer to the Web Frontend for user display

---

3. Data Flow Overview

Data Flow Step	Source	Destination	Data Type
PDF Upload	User (Frontend)	Flask Backend	PDF file

Text Extraction	PDF Processor	RAG Engine	Extracted text
Question Submission	User (Frontend)	Flask Backend	User question (text)
Context Retrieval & Answering	RAG Engine	Flask Backend	Generated answer (text)
Answer Display	Flask Backend	User (Frontend)	Answer (text)

---

4. Technology Stack

Layer/Function	Technology/Framework
Frontend	HTML/CSS/JavaScript
Backend/API	Python, Flask
PDF Processing	PyPDF2 or pdfplumber
RAG/QA Engine	LangChain, OpenAI API
Storage	Local filesystem or cloud

---

5. Scalability & Reliability

- **Scalability:**
  - Stateless backend enables horizontal scaling.
  - External AI services (OpenAI) handle computational load for answer generation.
  - Storage can be extended to cloud solutions for larger deployments.
- **Reliability & Security:**
  - Input validation and file type checks for uploads.
  - Temporary storage with regular cleanup to protect user data.
  - Secure API key management for OpenAI and LangChain integrations.

End of Document