PDF-QA-Application

High-Level Design (HLD)

1. Overview

The application enables users to upload PDF documents, ask questions regarding the content of these documents, and receive responses through NLP-based processing. It is a full-stack application with clearly defined frontend, backend, and storage mechanisms.

---

2. System Architecture Diagram

- **Frontend**: React.js provides the user interface for uploading PDFs and posing questions.
- **Backend**: FastAPI handles API requests, processes uploaded PDFs, and communicates with NLP libraries (LangChain) to generate answers.
- **Database**: PostgreSQL stores metadata about uploaded PDFs.
- **File Storage**: Local filesystem stores PDF files.

---

3. Key Components

1. **Frontend**
    - PDF Upload Interface
    - Questions Submission and Answer Display Interface
2. **Backend**
    - API Endpoints:
        - /upload: Handles PDF uploads.
        - /ask: Processes questions and generates answers.
    - File Handling and Text Extraction:
        - PyMuPDF for text extraction.
    - NLP Processing:
        - LangChain/.
3. **Storage**
    - PostgreSQL: Stores metadata (filename, upload time, etc.).
    - Local Filesystem: Stores the uploaded PDFs.
4. **NLP Engine**

- Handles semantic understanding of the document content to generate precise answers.

---

4. Functional Flow

1. **PDF Upload**:
   - User uploads a PDF → Frontend sends file to the /upload endpoint.
   - Backend extracts text and saves metadata and file.
2. **Ask Question**:
   - User enters a question → Frontend sends query to the /question endpoint.
   - Backend processes question using LangChain and returns an answer.
3. **Answer Display**:
   - Frontend receives the response and displays it to the user.

Low-Level Design (LLD)

1. Backend Modules

PDF Upload

- **API Endpoint**: /upload
  - Accepts PDF as a multipart/form-data request.
  - Validates file type and size.
  - Extracts text content using PyMuPDF.
  - Saves file to the local filesystem.
  - Stores metadata in the database.

Question Processing

- **API Endpoint**: /ask
  - Accepts a question and document ID.
  - Loads text content from the database or filesystem.
  - Processes question using LangChain/LLamaIndex.
  - Returns the generated answer.

---

2. Frontend Components

Upload Component

- File input field for PDF uploads.
- Handles file selection and submission to /upload.

Question-Answer Component

- Text input field for entering questions.
- Displays answers received from the backend.

API Integration

- **Axios**: Used for HTTP requests to the backend.
- **State Management**: Manages upload and question state.

---

3. Storage

- **Local Filesystem**: PDF files are stored in a designated folder (e.g., uploads/).
- **Database**: Metadata about uploaded PDFs is stored in PostgreSQL.

---

4. NLP Workflow

1. Text content from PDFs is pre-processed (tokenized, indexed).
2. LangChain processes the question against the indexed content.
3. Generates and returns an answer.

---

5. Security Measures

- Validate file types to prevent malicious uploads.
- Limit file size for uploads.
- Sanitize user inputs to prevent SQL injection and other attacks.
- Use HTTPS for secure communication.

---

6. Error Handling

- **Frontend**:
    - Show appropriate error messages for invalid file uploads or server errors.
- **Backend**:
    - Return HTTP status codes (e.g., 400 for invalid inputs, 500 for server errors).