High-Level Design (HLD)

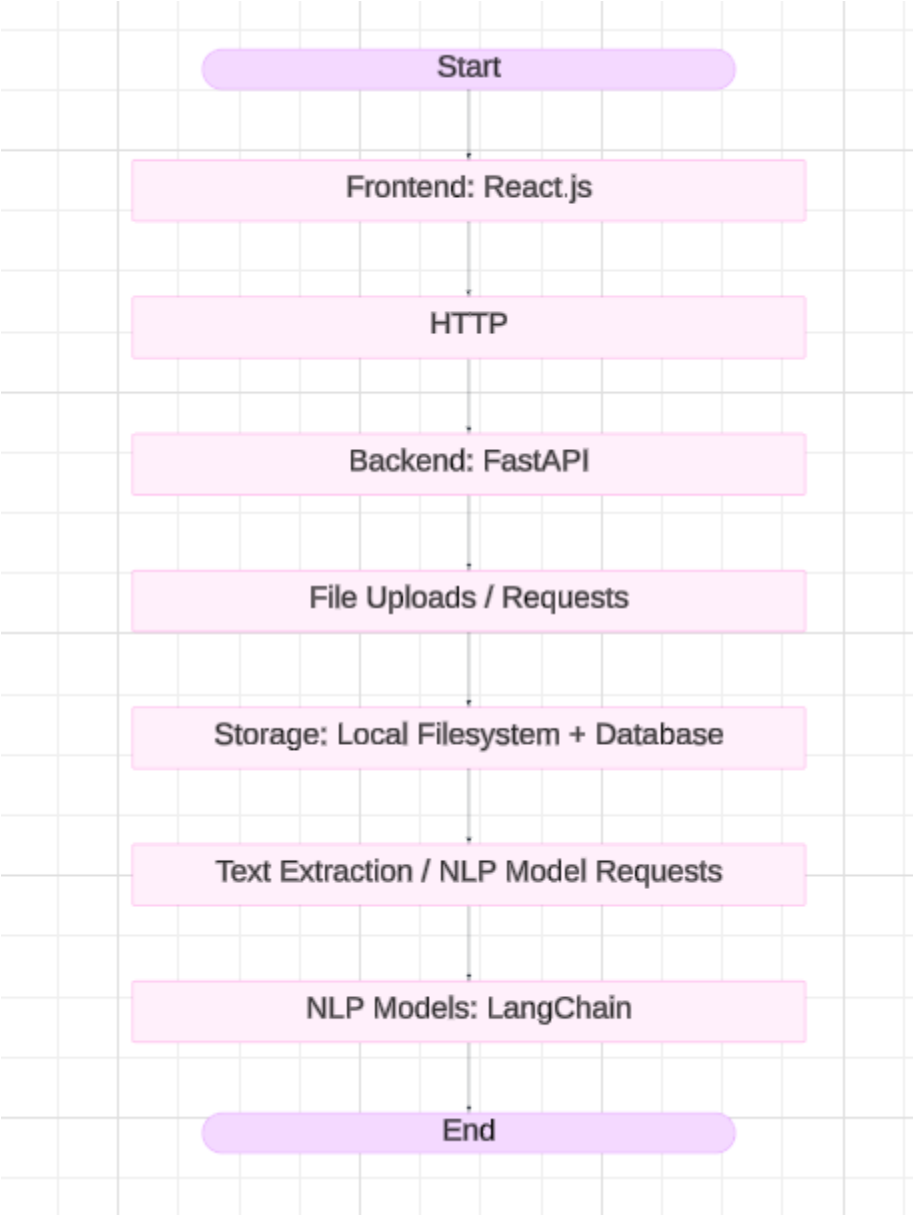
Objective

The system processes user-uploaded PDFs, extracts their content, and answers user questions about the content using natural language processing (NLP).

Components

1. **Frontend**:
    - **Technology**: React.js
    - **Responsibilities**:
        - Provides a user interface for uploading PDFs and asking questions.
        - Displays responses returned by the backend.
    - **Key Features**:
        - File upload functionality.
        - Question input and answer display.
2. **Backend**:
    - **Technology**: FastAPI
    - **Responsibilities**:
        - Handles file uploads and text extraction from PDFs.
        - Processes user questions and generate responses using NLP models.
    - **Key Features**:
        - PDF upload endpoint.
        - Question-answering endpoint integrated with LangChain for context-based answers.
3. **Storage**:
    - **Options**:
        - Local storage for uploaded files and extracted text.
        - SQLite for storing metadata (e.g., filenames, upload timestamps).
    - **Responsibilities**:
        - Persists uploaded PDF files and their extracted content.
        - Maintains records of processed files for quick retrieval.
4. **NLP Model Integration**:
    - **Technology**: LangChain, OpenAI API
    - **Responsibilities**:
        - Processes document content and user questions.
        - Generates accurate answers based on contextual understanding.

System Architecture Diagram



Low-Level Design (LLD)

Frontend

- **File Upload Component**:
  - Input: PDF file.
  - Logic: Converts the selected file into a `FormData` object and send it to the backend.
  - Endpoint: `/upload-file/`.
  - Output: Success/failure message based on the backend response.
- **Question Submission Component**:
  - Input: Question text and associated file name.
  - Logic: Send a `POST` request with the question and filename as parameters.
  - Endpoint: `/ask-question/`.
  - Output: Display the answer received from the backend.

Backend

- **Endpoints**:
  - `/upload-file/`:
    - Input: PDF file (multipart form).
    - Logic:
      - Reads the file.
      - Extracts text using `pdfplumber`.
      - Stores the extracted text in memory (or database).
    - Output: Success/failure message.
  - `/ask-question/`:
    - Input: Filename, Question text (form data).
    - Logic:
      - Retrieves the text associated with the file.
      - Uses LangChain to generate an answer from the context.
    - Output: Answer text or error message.
    -

Storage

- **Database**:
  - Schema:
    - `documents` table:
      - `id`: Primary key.
      - `filename`: Name of the uploaded file.
      - `upload_date`: Timestamp of upload.
      - `text_content`: Extracted text from the PDF.

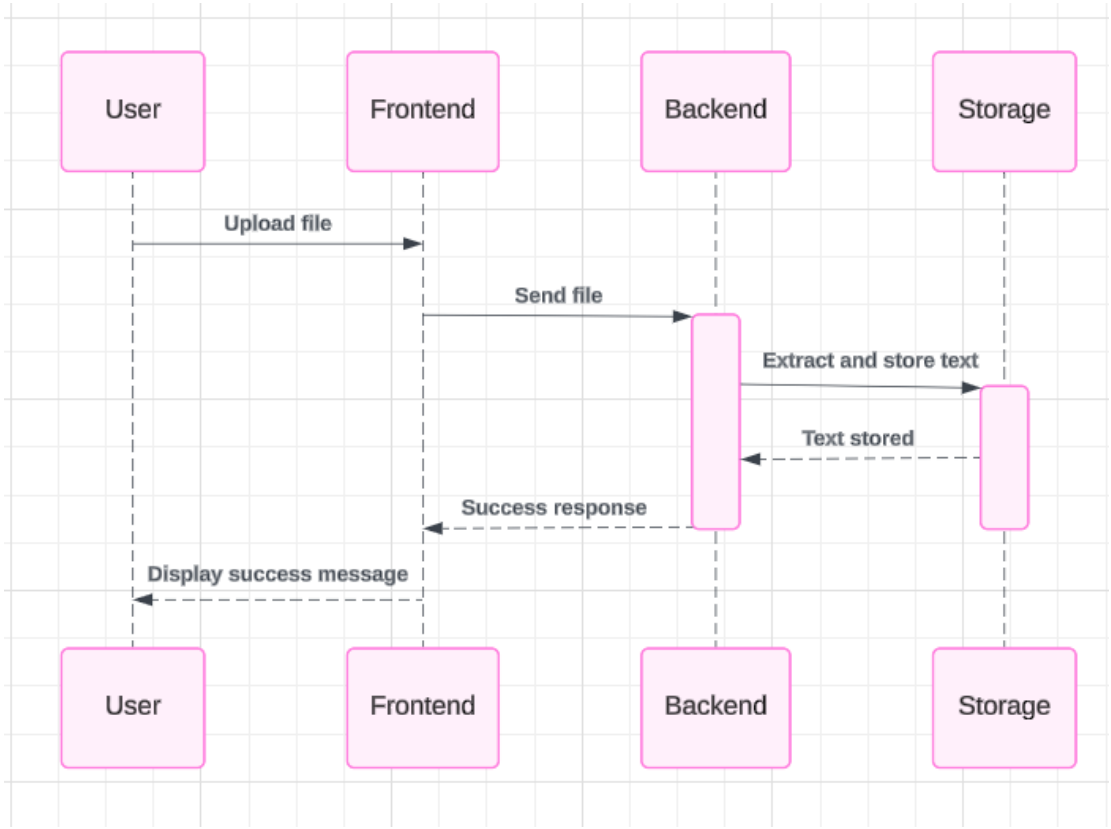
    - Queries:
        - Insert: Stores uploaded file metadata and text.
        - Retrieve: Fetches file text by filename.
- **Local Filesystem**:
    - Store raw PDF files.

NLP Processing

- **Input**: Extracted document text and user question.
- **Logic**:
    - Uses LangChain to process the document as context.
    - Queries the document context with the user's question.
- **Output**: NLP-generated answer.

Sequence Diagram

1. **File Upload Flow**:



2. **Question Submission Flow**: