Source Code Documentation

1. Backend

The backend is implemented using **FastAPI**, a modern Python framework for building APIs. It processes PDF uploads, extracts content, and answers user questions based on the extracted text.

Key Files

a. `main.py`

- **Purpose:** The entry point of the backend application. It defines the API endpoints and the business logic.
- **Core Components:**
  1. **Endpoints:**
     - **`/upload-file/` (POST)**:
       - Accepts PDF files from the frontend.
       - Extracts text using the `pdfplumber` library.
       - Stores the extracted content in an in-memory dictionary (`pdf_texts`).
     - **`/ask-question/` (POST)**:
       - Accepts a question and the filename.
       - Searches for sentences in the PDF content that match the question.
       - Returns a list of related sentences.
  2. **Middleware:**
     - **CORS**: Allows cross-origin requests to support communication between the React frontend and the FastAPI backend.
  3. **Helper Functions:**
     - Logic to parse PDF content into text.
     - Basic string matching for question-answering.

b. `requirements.txt`

- **Purpose:** Defines the Python libraries required for the backend.
- **Key Dependencies:**
  - `fastapi`: Framework for building the API.
  - `uvicorn`: ASGI server to run the FastAPI app.
  - `pdfplumber`: Library to extract text from PDF files.
  - `pydantic`: Data validation and settings management.

2. Frontend

The frontend is a **React.js** application. It provides a user interface for uploading files and asking questions.

Key Files

a. `App.js`

- **Purpose:** The main React component that sets up the app.
- **Core Features:**
  - Defines the structure of the application.
  - Handles navigation and integrates child components.

b. `FileUpload.js`

- **Purpose:** Component for uploading PDF files.
- **Core Features:**
  - Accepts PDF files and sends them to the `/upload-file/` backend endpoint.
  - Displays success or error messages based on server responses.

c. `QuestionForm.js`

- **Purpose:** Component for submitting questions related to uploaded PDFs.
- **Core Features:**
  - Sends a question and filename to the `/ask-question/` endpoint.
  - Displays the response, which includes sentences related to the question.

d. `index.js`

- **Purpose:** Entry point for the React application.
- **Core Features:**
  - Mounts the React app to the DOM.
  - Includes global application setup and rendering logic.

e. CSS and Supporting Files

- **`App.css` and `index.css`:** Provide styles for the application.
- **`logo.svg`:** Contains the logo asset for the frontend.
- **`reportWebVitals.js` and `setupTests.js`:** Utilities for performance monitoring and testing.

Interactions Between Backend and Frontend

1. **File Upload:**
   - The frontend uses `FileUpload.js` to send a PDF file to the backend's `/upload-file/` endpoint.
   - The backend processes the file, extracts its content, and stores it for future reference.
2. **Question Answering:**
   - The frontend uses `QuestionForm.js` to send a question and the filename to the backend's `/ask-question/` endpoint.
   - The backend searches the stored PDF content and returns related sentences, which are displayed in the frontend.

1. The user uploads a PDF through the **FileUpload** component in the frontend.
2. The backend processes the PDF and extracts its text.
3. The user submits a question through the **QuestionForm** component in the frontend.
4. The backend retrieves the extracted text, matches the question, and returns the relevant content.