Project Reference Document

Overview

This document provides a detailed overview of the technologies used and the steps to build the project, comprising a React + TypeScript frontend and a FastAPI backend. It is intended to help understand the stack and architecture of the project.

Tech Stack

Frontend

- **React**: Library for building the user interface.
- **TypeScript**: Adds static typing to JavaScript for better development experience.
- **Vite**: Development server and build tool for fast project setup.
- **Tailwind CSS**: Utility-first CSS framework for styling.
- **Lucide-React**: Icon library for UI components.
- **ESLint**: Tool for maintaining code quality and enforcing coding standards.
- **React Hooks**: Built-in state and lifecycle management for functional components.

Backend

- **FastAPI**: Web framework for building APIs quickly and efficiently.
- **Uvicorn**: ASGI server to run the FastAPI application.
- **PyPDF2**: For extracting text and metadata from PDFs.
- **Sentence-Transformers**: Library for generating embeddings for text queries and PDF content.
- **FAISS (CPU)**: Library for performing similarity searches between embeddings.
- **Pydantic**: For data validation and settings management.
- **Python-Multipart**: Handles file uploads in FastAPI.
- **Python-Dotenv**: Manages environment variables in `.env` files.

Building the Project

Frontend Implementation

1. **Initialize the Project**:

Create a React + TypeScript project using Vite:

```
npm create vite@latest my-project --template react-ts
```

2. **Install Dependencies**:

```
npm install tailwindcss lucide-react react-dom react
npm install -D eslint @vitejs/plugin-react
```

3. **Set Up Tailwind CSS**:

Initialize Tailwind configuration:

```
npx tailwindcss init
```

Add Tailwind to the `index.css` file:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

4. **Develop the UI**:
   Create components for:
   - File upload (PDFs).
   - Query input field.
   - Results display.
5. **Connect Frontend to Backend**:
   - Use `fetch` or `Axios` for API communication.
   - Handle responses from the backend to display relevant answers.

6. **Run the Development Server**:

```
npm run dev
```

Backend Implementation

1. **Set Up the Environment**:

Create a virtual environment:
 python -m venv venv
source venv/bin/activate

   ○

2. **Install Dependencies**:

Use the `requirements.txt` file:
 pip install -r requirements.txt

   ○

3. **Implement Core Functionality**:
   - **PDF Processing**:
     - Extract text from PDFs using `PyPDF2`.
   - **Embedding Generation**:
     - Use `sentence-transformers` to generate embeddings for:
       - PDF content.
       - User queries.
   - **Similarity Search**:
     - Store embeddings in a FAISS index and perform searches to retrieve matching results.
4. **Define FastAPI Endpoints**:
   - **Upload PDFs**:
     - Endpoint to accept and process uploaded files (`POST /upload`).
   - **Query Content**:
     - Endpoint to handle user queries and return matching results (`POST /query`).

Run the Server:

 uvicorn main:app --reload