pdf-annotator-full-stack

Problem Statement:

Build a full-stack React application that allows users to upload PDF files, highlight text within the document, and persist these highlights for later viewing. The application should support user login, saving uploaded PDFs and their annotations to a backend, and reloading them with previously saved highlights. PDF files should be stored locally on the server machine. Each file and its highlights should be uniquely tracked using UUIDs.

Objective:

Develop a full-stack PDF annotator application with the following capabilities:

Core Features

Frontend (React):

1. User Authentication
    - Implement registration and login using email/password.
    - Use JWT tokens for session handling.
2. PDF Upload and Viewer
    - Allow users to upload PDF files from their local system.
    - Display the uploaded PDF in-browser with pagination and zoom support.
3. Text Highlighting
    - Let users select and highlight text across any page in the PDF.
    - Each highlight should be stored with necessary metadata, including:
        - UUID of the PDF
        - Page number
        - Highlighted text
        - Position or bounding box
        - Optional: timestamp
4. Persisting Highlights
    - Save highlights in the backend database (MongoDB) associated with the user's account and the corresponding PDF UUID.
5. Restoring Highlights

    - When a previously uploaded PDF is re-opened, retrieve and display its associated highlights automatically.
6. My Library (Dashboard)
    - Show a list of previously uploaded PDFs for the logged-in user.
    - Allow users to open, rename, or delete files.

Backend (Node.js + Express):

1. Authentication APIs
    - Endpoints for login, signup, and token verification.
2. PDF Upload API
    - Accept and store uploaded PDF files on the server's local file system.
    - Generate and return a UUID for each uploaded PDF.
    - Save metadata like filename, user ID, and UUID in MongoDB.
3. Highlight API
    - Endpoints to create, retrieve, update, or delete highlights by PDF UUID and user.
    - Data is stored in MongoDB under the associated user and PDF record.
4. PDF Listing API
    - Fetch list of uploaded PDFs for a logged-in user.

Tech Stack & Tools:

- Frontend: React (with optional libraries like `react-pdf`, `pdfjs-dist`, `pdf-lib`)
- Backend: Node.js with Express
- Database: MongoDB
- Authentication: JWT-based login
- Storage: Local file system (PDFs saved on the machine where the backend is running)
- Identification: All PDF files should be uniquely identified using UUIDs

You may use libraries like `uuid`, `multer` (for handling file uploads), and `mongoose` for MongoDB integration. PDF rendering and text extraction can be handled using `pdfjs-dist` or `react-pdf`, as you prefer. Feel free to use any other library if the above doesn't satisfy the objective or if more comfortable using any other libraries.

Deliverables:

- A fully functional full-stack application with:
    - User login system
    - PDF upload and local storage
    - PDF viewing and highlighting
    - Highlight persistence and restoration
    - PDF library/dashboard per user
- A README that includes:
    - Overview of architecture and features
    - Setup instructions for running both frontend and backend locally
    - Sample `.env` or config setup (omit actual secrets)

Success Criteria:

- Each user can upload, annotate, and view only their own PDFs and highlights.
- Highlights are accurately saved and restored using PDF UUIDs.
- Uploaded files are correctly stored and managed on the server file system.
- Application behaves reliably across login sessions.

Optional / Bonus Features:

You may optionally implement the following advanced features to enhance the application:

- Search Functionality
    - Search for the full text of the PDF or annotations.
- Drawing Tools
    - Allow freehand annotations, arrows, and shapes in addition to text highlights.
- Custom Notes
    - Enable users to add comments or tags to each highlight.
- Cloud Storage Integration
    - Add integration with Google Drive or Dropbox for file import/export.
- Collaboration
    - Allow users to share PDFs with others (view-only or editable access).
- Text Extraction / NLP
    - Automatically summarize PDFs or extract key phrases using NLP libraries.
- Offline Access
    - Add offline capability using IndexedDB and service workers.

Delivery Method:

- Make use of GitHub to create repositories for the codebases.
- Ensure your repositories are publicly accessible.
- Share your repository link(frontend/backend) along with instructions on running the project on respective readme files.
- Email your repo links to ajay@deeref.co along with your resume.