Peer-to-Peer Book Sharing System

Documentation

[Done by Cynthia, Helin, Nakai and Hamna]

Introduction: -

The purpose of the Peer-to-Peer Book Sharing System is to make book exchanges easier within a community. Users can manage lending and borrowing, list books, make loan requests, and keep track of their reading habits. The system emphasizes user-friendly interaction and effective book management, and it is done utilizing file-based storage (JSON/CSV).

Objectives:-

- To develop a simple and easy-to-use book sharing platform.
- To make it easy for users to list, search, and borrow books.
- To put file-based storage into place for effective data administration.
- To allow alerts for updates and loan requests.
- To guarantee reliable input validation and error handling for a seamless user experience.

Features: -

1. Book Listing

- Users can **add, update, and delete** book details such as:
    - Title
    - Author
    - Genre
    - Condition
    - Availability status
- Users can **browse and search** for books by:
    - Title

    - Author
    - Genre
- To show their lending status, users may mark books as either available or not.

2. Borrowing and Lending

- Users can make loan requests for available books.
- Book owners have the option of accepting or rejecting loan requests.
- When a book is borrowed, the status is changed accordingly.
- When a book is returned, the status is changed to indicate availability.

3. User Features

- Users can establish and manage their own profiles.
- Users can review their personal lending and borrowing history.
- Users receive loan request notifications, which keep them up to date on borrowing and lending processes.

4. Currently Reading

- The "Currently Reading" page shows a table of books that the user is currently borrowing.
- Each book includes a Return button.
- Clicking the "Return" button removes the book from the "Currently Reading" list.

5. Notification

- The notification system is limited to book owners.
- Borrowers cannot change the book's availability status; only the owner can.

How Components Work Together: -

Frontend: Offers the user interface through which users can interact with the system.

With a visually appealing layout, users may add books, browse through pages, and reply to requests.

Backend: The logic and functionality are implemented by the backend.

Checks lending requests, book listings, user authentication, and notifications.

File-based storage guarantees that user, book, and loan transaction data is persistent. Writes and reads information from files to maintain system consistency.

The flow of interactions: The frontend, which interacts with users, talks to the backend services to perform its functions.

Through the file storage system, data is updated and retrieved, guaranteeing a coordinated flow.

Implementation: -

Helin (Frontend):

Used HTML, CSS, and JavaScript to create a user interface that is both responsive and appealing to look at. .

Task summary:

Used Figma to plan the design of the online application.

Made organized pages for:

1. Signup, Homepage, and Login
2. Books on Loan, My Inventory, Currently Reading, Requests
3. Icons, tables, and interactive buttons were styled to improve user experience.
4. made sure the navigation bar was responsive and styled.
5. Pagination was included to allow for dynamic book browsing.
6. For improved readability, user icons, placeholders, and layout changes were made.
7. For improved usage, practical icons and images were included.

Cynthia (Backend) :

Responsible for designing and implementing the backend logic of the system, ensuring smooth data flow between the client and server. The backend was developed using **JavaScript (Node.js/Express) with RESTful APIs** to manage user authentication, book inventory, and loan transactions.

- **Built RESTful APIs** to handle core functionalities:
  - **User Registration & Login** (Secure authentication)
  - **Book Management** (CRUD operations)
  - **Loan Requests & Approvals**
  - **Search & Filtering Mechanism**
- **Database Integration**: Connected the API with a database (**SQLite / PostgreSQL / MongoDB**, depending on the setup) to store user data and book details.
- **Optimized API Performance**: Implemented **efficient data queries** and caching strategies for faster response times.

Nakai (Backend):

Messaging Section:

- Ensured that once a loaner accepts a book request, the borrower and loaner can text each other.
- Developed a layout to search for usernames and enable chat functionality after request acceptance.

Notifications:

- Implemented a notification system to send and receive updates about loan requests.

Hamna (Backend for Currently reading page)

The "Currently Reading - Return Process" fetches borrowed books from book.json, arranges them in a table, and assigns a "Return" button to each one. When clicked, the button removes the book from the borrower's list and modifies the book.json file. Instead of automatically labeling the book as available, the owner gets a message and can determine whether or not it is available.

Made the PowerPoint presentation with a quick project overview, explaining the system's goal and characteristics. Include a slide outlining technical needs (such as user registration and notifications), followed by

contributions. Finish with a demo slide demonstrating how the system works and a conclusion.

Set Up

Springboot and its respective dependancies

Maven

Java17

Any IDE that can support Java, css,html and js

Conclusion: -

Within a community, the Peer-to-Peer Book Sharing System effectively facilitates book sharing. Book listings, loan transactions, and user interactions are all managed with its user-friendly interface and strong backend.

Results of the Project:

- Created a practical and easy-to-use book-sharing website.
- Enhanced ability to work together as a team.
- Future improvement topics were identified, including automating notifications and putting in place a database for increased scalability.