Library Management System

Frontend

The Library Management System is designed to allow **Librarians** and **Members** to manage books and their borrowing history. The system facilitates:

* Managing a collection of books.
* Recording book borrowing and returning events.
* Assigning different roles (Librarian and Member) to users, with different access controls.
* Tracking the borrowing history of books by members.

Features:

* Proper CRUD operation using REST API’s
* JWT based authentication in each protected endpoint.
* There are two roles in the system; LIBRARIAN and MEMBER
* Member can sign up either as LIBRARIAN and MEMBER using username and password
* Member can login using username/password and get JWT access token
* Librarian can add, update, and remove Books from the system
* Librarian can update, view, and remove Member from the system
* Librarian can view history of all the members(issue and return of books)
* Librarian can view deleted and active members.
* Membercan view, borrow, and return available Books
* Once a book is borrowed, its status will change to BORROWED
* Once a book is returned, its status will change to AVAILABLE
* Member can delete my own account.
* Member can view the history of books borrowed.

**Technical Stack**

**Frontend:**

* + **HTML, CSS, JavaScript used to create pages.**
  + **Hosting on Vercel**

**Page Flows:**

1. Login Page (login.html)
   1. Functionality: Allows users (librarians or members) to log into the system.
   2. API: POST /api/users/login
   3. Flow:
      1. The user enters their credentials and clicks "Login".
      2. A fetch request is made to the backend API with the login details.
      3. On successful authentication, a JWT token, username, userId and user role are saved in localStorage.
      4. The user is redirected to the book list (view\_books.html).
2. Signup Page (signup.html)
   1. Functionality: Allows new users to register as either a librarian or member.
   2. API: POST /api/users/signup
   3. Flow:
      1. The user enters their registration details and selects a role (librarian or member).
      2. A fetch request is made to the backend to create the account.
      3. On successful registration, the user is prompted to log in.
3. View Books Page (view\_books.html)
   1. Functionality: Displays the list of books available in the library. Librarians also have the ability to add new books.
   2. API: GET /api/books/
   3. Flow:
      1. On page load, a fetch request is made to the backend to retrieve the list of books.
      2. Each book is displayed as a clickable div containing the book’s title and author.
      3. If the user is a librarian (role from localStorage), an "Add Book" button is shown on the page.
      4. When a book is clicked, the user is taken to the Book Details page (book\_details.html).
4. Librarian Specific: Add Books Page (add\_books.html)
   1. Functionality: The Add Books Page is accessible only to users with the librarian role. This page allows librarians to add new books to the library system by providing relevant book information.
   2. API: POST /api/books/add.
   3. Flow:
      1. Book Information Form:
         1. The page includes a form with input fields for entering the book's title, author. These fields allow the librarian to provide the required details to create a new book entry.
      2. Submit Form:
         1. Once the librarian fills out the form, they can click the "Add Book" button to submit the data. This triggers an API call to the backend to store the book information in the database.
      3. Upon successful addition of the book, a confirmation message is displayed, and the form is cleared and the person is navigated to the view\_books.html
5. Profile Page (settings.html)
   1. Functionality:
      1. The Profile Page allows users (both members and librarians) to view their account details and borrow history. This page serves different functionality for regular members and librarians:
      2. Members can see their profile details, borrow history, and have an option to delete their account.
      3. Librarians have the ability to view their profile and can navigate to the "View Members" page to manage users.
   2. API
      1. GET /api/users/profile
      2. DELETE /api/users/delete/self
   3. Flow:
      1. When the page loads, the user's information (like name, email, and role) is retrieved from local storage or through an API call using the stored token.
      2. For librarians, an additional button is displayed allowing them to navigate to the "View Members" page.
      3. For members, their book borrowing history is fetched via an API call and displayed in a list format.
      4. Members can see a "Delete Account" button, which, when clicked, will prompt the user for confirmation to delete their account.
      5. If successful, the user's session is cleared from local storage, and they are redirected to the login page.
6. Book Details Page (book\_details.html)
   1. Functionality: Displays detailed information about a specific book and provides options to borrow/return (for members) or update/delete (for librarians).
   2. API:
      1. GET /api/books/:id,
      2. POST /api/books/borrow/:bookId/user/:userId,
      3. DELETE /api/books/delete/:bookId
   3. Flow:
      1. On page load, the book ID is retrieved from the URL parameters, and a fetch request is made to load the book's details.
      2. If the user is a librarian, options to update or delete the book are provided.
      3. If the user is a member, options to borrow or return the book are shown.
      4. The "Go Back" link redirects the user to the book list (view\_books.html).
      5. If the person is LIBRARIAN, they will be shown a button Update Book and Delete Book. When clicked “Update Book” will navigate to the update\_book.html page, where the LIBRARIAN can update the book details.
      6. When clicked “Delete Book” will send a API delete request to backend and upon successful deletion, the LIBRARIAN will be redirected to the view\_books.html page.
7. Librarian Specific: Update Book Page (update\_book.html)
   1. Functionality: he Add Books Page is accessible only to users with the librarian role. This page allows librarians to update the books to the library system by providing relevant book information.
   2. API: POST /api/books/update/:bookId.
   3. Flow:
      1. Book Information Form:
         1. The page includes a form with input fields for entering the book's title, author. These fields allow the librarian to provide the required details to create a new book entry.
         2. The bookId will be recorded from the params of the previous redirected page when clicked on the Update Book button.
      2. Submit Form:
         1. Once the librarian fills out the form, they can click the "Add Book" button to submit the data. This triggers an API call to the backend to store the book information in the database.
      3. Upon successful addition of the book, a confirmation message is displayed, and the form is cleared and the user will be navigated to the view\_books.html page.
8. View Members Page (view\_members.html)
   1. Functionality: Allows librarians to view active and inactive members. The members are displayed as clickable divs.
   2. API:
      1. GET /api/users/activeMembers,
      2. GET /api/users/deletedMembers
   3. Flow:
      1. On page load, active members are displayed by default.
      2. A toggle button allows the librarian to switch between active and inactive members.
      3. Clicking a member redirects to the Member Details page (member\_details.html).
9. Librarian Specific: Member Details Page (member\_details.html)
   1. Functionality: Displays the profile details and borrowing history of the selected member.
   2. API:
      1. GET /api/users/member/:userId,
      2. DELETE /api/users/delete/:userId
   3. Flow:
      1. On page load, the member ID is retrieved from the URL parameters, and a fetch request is made to load the member's profile and borrow history.
      2. If the librarian navigates from the active members view, they are given an option to delete the member.
      3. If the librarian is viewing inactive members, the delete button is not shown.

**JavaScript Files:**

All these html files has corresponding javascript files, which handles the fetch functions and other necessary operations.

Logout:

* The logout function (logout() in header.js) clears the token and role from localStorage and redirects the user to the login page.

CONCLUSION:

The frontend of the Library Management System provides an intuitive and role-based interface for managing books and members. It is built on a modular structure, where each page has a corresponding JavaScript file that handles the specific functionalities. Interaction with the backend is done using fetch API, and user authentication is managed via JWT tokens stored in localStorage. This ensures a secure and seamless experience for both librarians and members.