LIBRARY MANAGEMENT SYSTEM

TECH STACK

- HTML
- CSS
- JAVASCRIPT
- Firebase
- Google Api

FOLDERS

- Assets
- Books
- Fine Amount
- Journals
- Login
- Message

WORKING

LOGIN

It is a simple login and registration form with Firebase authentication integration. It allows users to log in with their email and password or register a new account. Here's an explanation of how it works:

1. HTML Structure
    - The HTML code defines a login and registration form.
    - There are two main sections: login-div and register-div.
    - Each section contains input fields for email and password, along with buttons for login, registration, and switching between login and registration forms.
    - There is also a result-box section that displays the result of login or registration attempts.

2. CSS Styling
    - The CSS code provides the styling for the form, including background gradients, box shadows, button styles, and animations.
    - It makes the form visually appealing and responsive.

3. JavaScript Code
    - The JavaScript code imports the necessary Firebase modules and initializes the Firebase app using the provided configuration.
    - It retrieves the authentication object (auth) from the Firebase app.
    - Event listeners are added to the login, registration, and logout buttons to handle user actions.

4. Login and Registration Logic
    - When the login button is clicked, the code retrieves the entered email and password values.
    - It then calls the signInWithEmailAndPassword function from the Firebase auth module, passing the email and password as arguments.
    - If the login is successful, the user is redirected to Home page, and a Library management System Home view will be visible. If there is an error during login, an error message is displayed.
    - When the registration button is clicked, the code retrieves the entered email and password values.
    - It then calls the createUserWithEmailAndPassword function from the Firebase auth module, passing the email and password as arguments.If the registration is successful, a success message is displayed. If there is an error during registration, an error message is displayed.

5. Logout Logic
    - When the logout button is clicked, the code calls the signOut function from the Firebase auth module.
    - If the logout is successful, the result box and login form are hidden, and the user is shown the login form again.
    - If there is an error during logout, an error message is displayed.

This code provides a basic login and registration functionality using Firebase authentication. It demonstrates how to integrate Firebase with a web application to handle user authentication securely.

HOME

This is a working model of Home Page in Library Management System. It consists of HTML, CSS, and JavaScript code.

1. HTML Structure
   - HTML includes the necessary meta tags, title, external stylesheets (swiper-bundle.min.css, style.css), and script files (script.js, swiper-bundle.min.js).
   - After login it will redirect to Home page
   - The content of the webpage is wrapped inside a container div.
   - The navigation bar is created using the <nav> element. It contains a logo on the left side and a sidebar menu on the right side. The sidebar menu is initially hidden and can be toggled by clicking on the menu icon (represented by the bx-menu class). The sidebar contains various links to different sections of the LMS, such as Home, Books, Journals, Fine Amount, and Messages. There is also a "Logout" link at the bottom of the sidebar.
   - The main section of the webpage is a swiper carousel (implemented using the Swiper library). It displays a series of slides with images and captions. The images are placed inside swiper-slide elements. Each slide contains an image and a caption with the text "LIBRARY MANAGEMENT SYSTEM" and an "EXPLORE" button. The swiper-pagination element represents the pagination bullets at the bottom of the carousel, and the swiper-button-next and swiper-button-prev elements are the navigation buttons for moving to the next and previous slides.

2. CSS Styling
   - The sidebar is initially hidden off-screen and slides in from the left side when the menu icon is clicked (by applying the "open" class to the nav element).
   - The images in the carousel are set to cover the entire slide area, and the caption text is centered vertically and horizontally.
   - The navigation buttons and pagination bullets are styled to match the overall design.

3. Javascript Code
   - The JavaScript code includes event listeners to handle the functionality of the sidebar menu and the carousel.
   - When the menu icon is clicked, the "open" class is toggled on the nav element, which triggers the transition animation to show or hide the sidebar.
   - The carousel is set to automatically slide to the next image every 2 seconds (using the autoSlide function and setInterval). The updateClick function is called when the next or previous buttons are clicked, updating the imageIndex and sliding the carousel accordingly.
   - The autoSlide function is called again to continue the automatic sliding after a button click. Additionally, the interval is cleared and restarted when the mouse is over the carousel (to pause the automatic sliding) and when the mouse leaves the carousel (to resume the automatic sliding).

This working model demonstrates a responsive webpage layout with a navigation bar, sidebar menu, and an interactive carousel for displaying images and captions.

BOOKS

Books section allows users to search for books using the Google Books API and displays the search results dynamically on the page. Here's an working model:

1. HTML Structure:
   - It includes a navigation bar, a search input field, and a container to display the search results.

2. CSS Styling:
   - The CSS styles define the appearance of the web page, including the navigation bar, search input field, and search results container.

3. JavaScript Functionality:
   - The first part of the JavaScript code selects the necessary HTML elements and sets up event listeners. It selects the navigation bar, menu buttons, and overlay element. When the menu buttons or overlay are clicked, it toggles the "open" class on the navigation bar, which controls the sidebar's visibility.
   - The code then sets up an event listener for the search button. When the search button is clicked, it triggers an AJAX request to the Google Books API with the search query entered by the user. If the request is successful and returns results, the code displays the search results dynamically on the page.
   - The displayResults function receives the response from the API and iterates over the items in the response to extract relevant information such as book title, author, publisher, book link, and book image. It formats the information into HTML cards and appends them to the outputList element.
   - The formatOutput function takes the extracted book information and formats it into an HTML card structure. It also includes a link to view more details about the book.
   - The displayError function is called if the search query is empty, displaying an alert to the user.

It allows users to search for books and view the search results.

JOURNALS

Filtering Books:

- The user can enter values in the filter input fields (Title, Author, Subject, Publish Date) to filter the list of books.
- As the user types in the input fields, an event listener listens for changes and triggers the renderBooks function.
- The renderBooks function filters the books array based on the values entered in the filter input fields.
- It then renders the filtered books by creating HTML elements for each book and appending them to the booksContainer element.

Pagination:

- The code implements pagination to display a limited number of books per page(10).
- The renderBooks function calculates the total number of pages based on the filtered books and the desired number of items per page.
- It determines the range of books to display based on the current page.
- It generates pagination buttons dynamically based on the total number of pages.
- The user can click on the pagination buttons to navigate to different pages and view different sets of books.

Requesting Books:

- Each book card generated by the renderBooks function has a "Request Book" button associated with it.
- When the user clicks on the "Request Book" button, a success message is displayed using the SweetAlert library.
- The success message can be customized or replaced with an actual functionality to handle book requests.

The functionality of requesting books and handling the requests is not fully implemented in this Library Management System. It displays a success message as a placeholder.

It also has a basic functionality such as a messaging option where users can enter their queries in message page, it sends the queries to the admin's email address using Formspree. Additionally, a user interface for a payment gateway has been created.

In the future, there are plans to implement backend functionality for the payment gateway.