Online Library Management System

Technology Stack

The technology stack for this application encompasses MongoDB for database management, Express.js for the backend server, React.js for the frontend user interfaces, and Node.js as the runtime environment. Mongoose, an Object Data Modeling (ODM) library, is employed for facilitating interactions between the application and the MongoDB database.

Backend Development:

The backend architecture is based on Express.js, providing a robust and scalable server-side logic. MongoDB is employed for data persistence, and Mongoose serves as the ODM for simplified interactions with the database.

APIs Endpoints

APIs endpoint For Book Management:

1. Create Book:

- Endpoint: POST /create
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Check if the book already exists based on the name and author.
    - If not, create a new book and return it.
    - Handles errors appropriately.

2. Update Book Availability:

- Endpoint: PATCH /:bookId
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Updates the availability status of a book based on the provided bookId.
    - Returns the updated book.
    - Handles errors, such as if the book is not found.

3. Delete Book:

- Endpoint: DELETE /:bookId
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Deletes a book based on the provided bookId.

    - Returns the deleted book.
    - Handles errors, such as if the book is not found.

4. Fetch All Books:

- Endpoint: GET /
- Functionality:
    - Fetches all books in the database.
    - Returns the list of books.
    - Handles errors, such as if there are no books.

APIs endpoint For Transaction Management:

1. Add Transaction:

- Endpoint: POST /add
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Adds a new transaction based on user email, book name, author, transaction type, and due date.
    - Retrieves user and book IDs from the database.
    - Returns the newly created transaction.
    - Handles errors, such as failures in transaction creation.

2. Update Transaction Type:

- Endpoint: PATCH /:transactionId
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Updates the transaction type (borrowed/returned) based on the provided transactionId.
    - Returns the updated transaction.
    - Handles errors, such as if the transaction is not found.

3. Delete Transaction:

- Endpoint: DELETE /:transactionId
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Deletes a transaction based on the provided transactionId.
    - Returns the deleted transaction.

    - Handles errors, such as if the transaction is not found.

4. Get All Transactions:

- Endpoint: GET /
- Middleware: Requires authentication.
- Functionality:
    - Fetches all transactions in the system.
    - Returns the list of transactions.
    - Handles errors, such as if there are no transactions.

5. Get User Transactions:

- Endpoint: GET /users/:userId/transactions
- Middleware: Requires authentication.
- Functionality:
    - Fetches all transactions associated with a specific user based on the provided userId.
    - Returns the list of user transactions.
    - Handles errors, such as if the user or transactions are not found.

6. Get Book Transactions:

- Endpoint: GET /books/:bookId/transactions
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Fetches all transactions associated with a specific book based on the provided bookId.
    - Returns the list of book transactions.
    - Handles errors, such as if the book or transactions are not found.

APIs endpoint For User Management:

1. Login User:

- Endpoint: POST /login
- Functionality:
    - Validates user credentials (name and email) for login.
    - Generates a token upon successful login.
    - Returns user details and token.
    - Handles errors, such as invalid credentials.

2. Add User:

- Endpoint: POST /add
- Middleware: Requires authentication and admin privileges.
- Functionality:
  - Adds a new user based on provided details.
  - Checks if the user already exists.
  - Returns user details and token upon successful creation.
  - Handles errors, such as existing user.

3. Delete User:

- Endpoint: DELETE /:userId
- Middleware: Requires authentication and admin privileges.
- Functionality:
  - Deletes a user based on the provided userId.
  - Returns the deleted user.
  - Handles errors, such as if the user is not found.

4. Get User Details:

- Endpoint: GET /:userId
- Middleware: Requires authentication.
- Functionality:
  - Fetches details of a user based on the provided userId.
  - Returns user details.
  - Handles errors, such as if the user is not found.

6. User Update:

- Endpoint: PUT /update
- Middleware: Requires authentication.
- Functionality:
  - Updates user details based on the authenticated user.
  - Returns updated user details and token.
  - Handles errors, if any.

7. Fetch All Users:

- Endpoint: GET /
- Functionality:

    - Fetches details of all users in the system.
    - Returns the list of users.
    - Handles errors, such as if there are no users.

APIs endpoint For Admin Management:

1. Register Admin User:

- Endpoint: POST /register
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Registers a new admin user with provided details.
    - Checks if the admin user already exists.
    - Returns admin user details and token upon successful registration.
    - Handles errors, such as existing admin user.

2. Admin User Login:

- Endpoint: POST /login
- Functionality:
    - Validates admin user credentials (username and password) for login.
    - Generates a token upon successful login.
    - Returns admin user details and token.
    - Handles errors, such as invalid credentials.

3. Admin User Update:

- Endpoint: PUT /update
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Updates admin user details based on the authenticated admin user.
    - Returns updated admin user details and token.
    - Handles errors, if any.

4. Delete Admin User:

- Endpoint: DELETE /:adminId
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Deletes an admin user based on the provided adminId.
    - Returns the deleted admin user.

        - Handles errors, such as if the admin user is not found.

5. Fetch All Admin Users:

- Endpoint: GET /
- Functionality:
    - Fetches details of all admin users in the system.
    - Returns the list of admin users.
    - Handles errors, such as if there are no admin users.

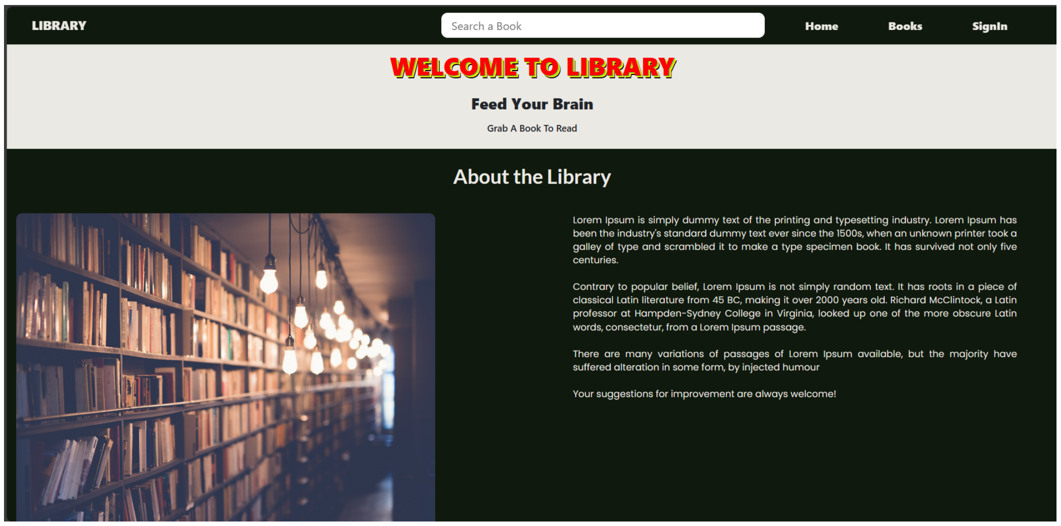
6. Get Admin User Details:

- Endpoint: GET /:Id
- Middleware: Requires authentication and admin privileges.
- Functionality:
    - Fetches details of a specific admin user based on the provided Id.
    - Returns admin user details.
    - Handles errors, such as if the admin user is not found.

These endpoints leverage express-async-handler for handling asynchronous operations, and appropriate status codes and error messages are implemented for each scenario.

Frontend Development

The landing page introduces our library and showcases all the available books. Anyone can visit this page. If you want to borrow books or access personalized features, you can sign in using the provided option.



WELCOME TO LIBRARY

Feed Your Brain

Grab A Book To Read

About the Library

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries.

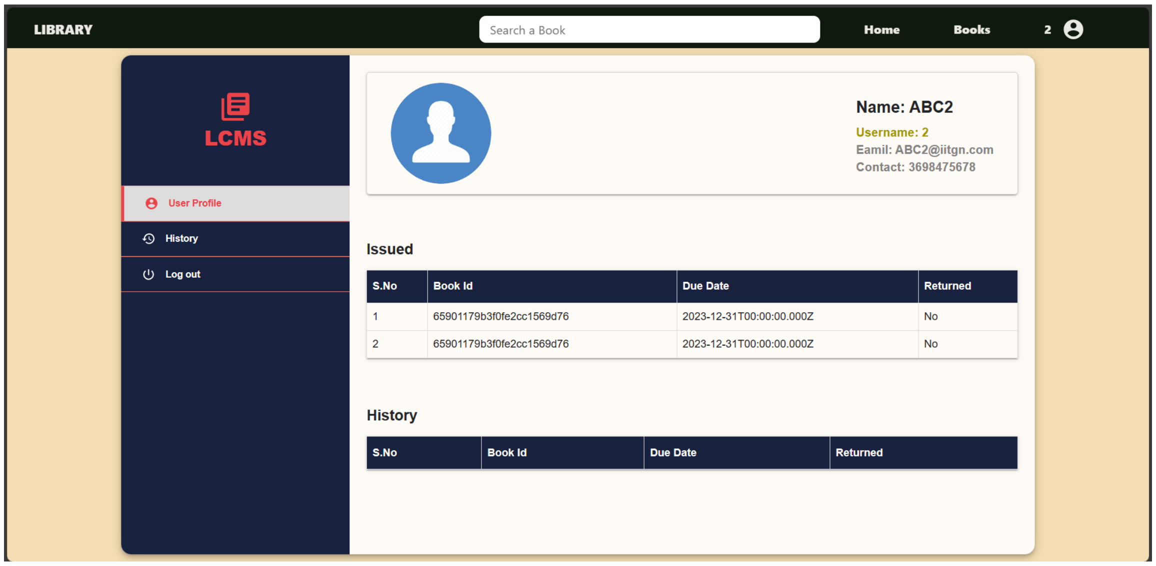
Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage.

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour

Your suggestions for improvement are always welcome!

Figure1: - Landing Page

Once a user signs in, they are directed to their Member Dashboard. Here, they can view their user details and see a record of all their transactions.



Name: ABC2
Username: 2
Eamil: ABC2@iitgn.com
Contact: 3698475678

Issued

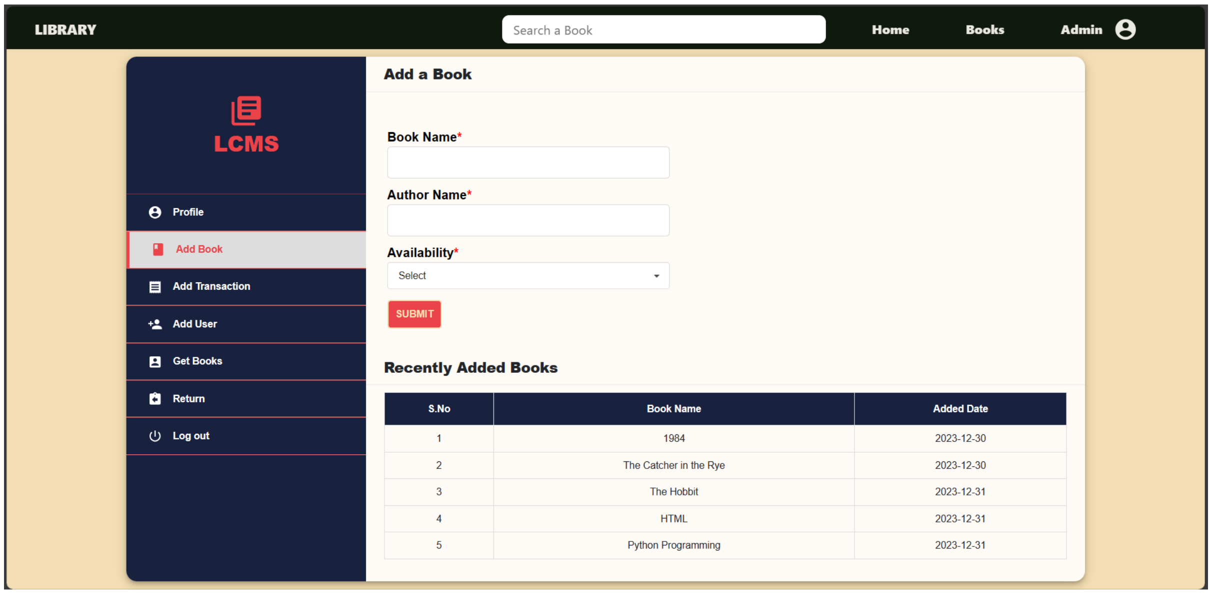
S.No	Book Id	Due Date	Returned
1	65901179b3f0fe2cc1569d76	2023-12-31T00:00:00.000Z	No
2	65901179b3f0fe2cc1569d76	2023-12-31T00:00:00.000Z	No

History

S.No	Book Id	Due Date	Returned

Figure2:- Member/User Dashboard

If a user signs in with admin credentials, they are directed to the Admin Dashboard. This dashboard provides access to a wide range of functionalities. Admins can add new books, users, and other administrators. They also have the ability to make transactions and edit book details.



Add a Book

Book Name*

Author Name*

Availability*
Select

Recently Added Books

S.No	Book Name	Added Date
1	1984	2023-12-30
2	The Catcher in the Rye	2023-12-30
3	The Hobbit	2023-12-31
4	HTML	2023-12-31
5	Python Programming	2023-12-31

Figure3:- Admin Dashboard