# MERN E-Library Project Documentation

## Table of Contents

## 1. Introduction

The MERN E-Library project is an online platform designed to provide users with access to a vast collection of Modules on various topics related to web development.

Leveraging the MERN (MongoDB, Express.js, React.js, Node.js) stack, the application offers an intuitive and user-friendly interface for browsing, searching, and interacting With modules.

## 2. Project Overview

The goal of the MERN E-Library project is to create an interactive and educational platform where users can explore and learn about different aspects of web development. The project aims to provide a comprehensive library of Modules covering topics such as HTML, CSS, JavaScript, React, Node.js, and MongoDB.

## 3. Features

- User authentication and authorization
- Browse Modules by category
- Search for Modules by title or category
- View detailed information about each module
- Add Modules to favorites and manage favorites list
- Leave reviews and ratings for Modules
- Bookmark pages within Modules for future reference
- Admin panel for managing Modules, categories, and users

## 4. Technical Architecture

The MERN E-Library project follows a client-server architecture, with the frontend built using React.js and the backend implemented using Node.js and Express.js. MongoDB is used as the database to store modules data.

## 5. Project Structure

## User Structure:-

***Home Page:-**

Header Component :- Logo, Browse Module Button,Home Button**,** Login/Logout Button, Contact button.

Body Component :- My Modules Component, Favourite Modules Component, User Contribution

***Browse Module Component** :-

Header Component :- Logo, Search bar, Home Button

Body Component :- HTML module, CSS module, JS module, React module, Node module, Mongo db module

##Admin Structure:-

***Home Page:-**

***Header Component:-** Logo, Home Button, Login/Logout Button
***Body Component :-** Current Modules, Add new Modules.

***Current Modules Component:-** HTML module, CSS module, JS module, React module, Node module, Mongo db module

***Add Modules Component:-**

***Header Component**:- Home Button, Modules Button
***Body Component**:- Form Component
***Form Component** :- Topic Input Box, Sub Topic Input Box, Add files Button, Upload Button.

- Frontend: React.js

- Backend: Node.js, Express.js, MongoDB, Mongoose

- User authentication: JSON Web Tokens (JWT)

- Deployment: Heroku (backend), Netlify (frontend)

## 6. User Roles and Permissions

- **User**: Can browse and search for Modules, view b details, add Modules to
favorites, leave reviews and ratings, and bookmark pages.

- **Admin:** In addition to user capabilities, admins can manage Modules, categories,
and user accounts through the admin panel.

## 7. Database Schema

The MongoDB database schema consists of a `Modules` collection to store book data.

Each book document includes fields such as `title`, `author`, `category`, `description`, `reviews`, `ratings`, etc.

```javascript

{

  title: String,
author: String,
category: String,
description: String,
  // Additional fields for reviews, ratings, etc.
}
```

## 8. API Documentation

### Book Routes:

- `GET /api/Modules`: Get all Module
- `GET /api/Modules/:id`: Get book by ID
- `POST /api/Modules`: Add new book
- `PUT /api/Modules/:id`: Update book by ID
- `DELETE /api/Modules/:id`: Delete book by ID

### User Routes:

- `POST /api/users/register`: Register new user
- `POST /api/users/login`: User login
- `GET /api/users/profile`: Get user profile
- `PUT /api/users/profile`: Update user profile

## 9. Frontend Components

- **Homepage**: Displays featured Modules, categories, and search bar.

- **Book List**: Renders a list of Modules based on category or search results.

- **Book Details**: Shows detailed information about a selected book, including reviews and ratings.

- **User Authentication**: Forms for user registration, login, and profile management.

- **Admin Panel**: Interface for admin users to manage Modules, categories, and users.

## 10. Deployment

- **Backend**: Deployed to Heroku platform as a Node.js application.
- **Frontend**: Hosted on Netlify as a static site, with API requests directed to the backend Heroku app.

## 11. Future Enhancements

- Implement more robust user authentication and authorization mechanisms.

- Enhance the admin panel with additional features such as analytics and reporting.

- Introduce advanced search functionalities, including filters and sorting options.

- Expand the library with a wider range of book categories and topics.

- Implement real-time notifications for user interactions such as reviews and ratings.

## 12. Conclusion

The MERN E-Library project provides a scalable and feature-rich platform for users to explore and learn about web development topics. With its intuitive user interface, comprehensive book collection, and interactive features, the application aims to empower users to enhance their skills and knowledge in the field of web development.

Logic Flow for the MERN E-Library Project

User Interaction Flow:

1. **User Visits the E-Library Website**:
   - User navigates to the E-Library website using a web browser.
2. **Homepage Display**:
   - The homepage of the E-Library is displayed, featuring sections for different categories such as HTML, CSS, JavaScript, React, Node.js, and MongoDB.
3. **Browsing Modules**:
   - User can browse Modules within each category by clicking on the respective category section.
   - Alternatively, the user can use the search functionality to find specific Modules by title, author, or category.
4. **Viewing Book Details**:
   - Upon selecting a book, the user can view detailed information about the book, including title, author, description, and possibly an image.
   - If available, the user can also view reviews and ratings submitted by other users.
5. **Filtering and Searching**:
   - User can filter Modules by category using category-specific navigation links or dropdown menus.

- User can also perform a search by entering keywords into the search bar, which dynamically updates the displayed Modules based on the search query.

6. **User Authentication (Optional)**:
    - If user authentication is implemented, the user may be prompted to log in or register for an account to access certain features such as saving favorites, leaving reviews, or accessing personalized recommendations.
7. **Interacting with Modules**:
    - User can interact with Modules by adding them to favorites, leaving reviews, rating Modules, or bookmarking pages for future reference.
    - For authenticated users, these interactions may be tied to their user account for persistence across sessions.

Admin Interaction Flow :

1. **Admin Panel Access**:
    - Admin users can access the admin panel by logging in with their admin credentials.
2. **Managing Modules**:
    - Within the admin panel, admins can perform CRUD operations on Modules, including adding new Modules, editing existing Modules, and deleting Modules.
    - Admins can also manage book categories and perform other administrative tasks related to book management.
3. **User Management**:
    - Admins may have the ability to manage user accounts, including creating new accounts, modifying user permissions, and deactivating or deleting user accounts if necessary.

Technical Flow:

1. **Frontend-Backend Communication**:
    - When the user interacts with the frontend interface, such as browsing Modules or submitting search queries, the frontend sends HTTP requests to the backend server.
    - These requests contain relevant data such as search keywords, selected categories, or user authentication tokens.

2. **Backend Processing**:
    - The backend server receives incoming HTTP requests and processes them by querying the MongoDB database, performing necessary operations such as filtering Modules, fetching book details, or updating user information.
    - If user authentication is implemented, the backend verifies user credentials and permissions before processing requests that require authentication.
3. **Database Interaction**:
    - The backend interacts with the MongoDB database to retrieve or update relevant data, such as fetching book information, storing user preferences, or updating book reviews and ratings.
4. **Response Generation**:
    - After processing the request and interacting with the database, the backend server generates an appropriate response, which may include data to be displayed on the frontend interface or status messages indicating the outcome of the request.
5. **Frontend Rendering**:
    - The frontend receives the response from the backend and updates the user interface accordingly, displaying relevant book information, search results, or error messages as necessary.
6. **User Feedback and Interaction**:
    - Throughout the process, the user interacts with the frontend interface, providing input, viewing results, and navigating through different sections of the E-Library application.