

NAAN MUDHALVAN PROJECT

MERN STACK BY MONGO DB

Project Title: BOOK STORE

Team Members:

ITHA VAISHNAVI SANJUKTHA- 113321243015

HARINI AM – 113321243013

HARSHITHA SHIVANI T - 113321243014

JEROMEY A - 113321243016

AI & DS (FINAL YEAR)

**VELAMMAL INSTITUTE OF
AND TECHNOLOGY-
PONNERI
CHENNAI - 601204**

Index

- **Project Overview**
- **Objectives**
- **Technology Stack**
- **System Requirements**
- **Features**
- **Project Architecture**
- **Installation and Setup**
- **Workflow and Usage**
- **Folder Structure**
- **ER Diagram**
- **Challenges Faced**
- **Future Enhancements**
- **Project Implementation & Execution**
- **Conclusion**

The Online Book Store project is a full-stack web application built using the MERN stack (MongoDB, Express, React, Node.js). It provides a platform for users to browse, search, purchase, and manage books online. Users can create accounts, add books to their cart, and complete purchases through secure payment integration (e.g., Stripe or PayPal). Admins have access to a dashboard where they can manage book inventory, process orders, and oversee users. The application includes features such as user authentication with JWT, responsive design, and real-time updates for orders and book availability. The project leverages MongoDB for data storage, React for the frontend, and Node.js/Express for the backend, offering a seamless, interactive experience for both customers and administrators.

2. Objectives

The primary objectives of an Book Store are:

1. **Seamless Shopping Experience:**
To provide users with a smooth and user-friendly platform to browse, search, and purchase books online with ease. This includes features like advanced search filters, sorting options, and detailed book pages.
2. **Secure User Authentication:**
To ensure that users can securely create accounts, log in, and manage their profiles. Authentication mechanisms like JWT (JSON Web Tokens) are used to protect user data and provide a secure login experience.
3. **Efficient Book Management:**
For admins to easily manage and update the book inventory, including adding new books, updating details (e.g., price, description), and removing unavailable books from the catalog.
4. **Order Management and Tracking:**
To allow users to place, track, and manage their orders with real-time updates on order status (e.g., processing, shipped, delivered). Admins should also be able to manage order processing and view order details.
5. **Payment Integration:**
To securely process payments using services like Stripe or PayPal, allowing users to make purchases with confidence and ensuring smooth payment transactions.
6. **Responsive Design:**
To provide an optimized and seamless experience across all devices, including desktops, tablets, and smartphones, ensuring users can shop anytime and anywhere.
7. **User Reviews and Ratings:**
To enable users to leave reviews and rate books, helping others make informed purchasing decisions based on community feedback.
8. **Customer Support:**
To offer easy ways for customers to contact support for inquiries, issues with orders, or other questions, creating a reliable customer service experience.

9. Scalability and Performance:

To ensure the platform can handle growth in terms of users, books, and transactions, and perform well under heavy traffic conditions.

Overall, the key objective is to create a convenient, secure, and engaging platform for both customers and administrators in the world of online book retail.

3. Technology Stack

The Book Store is built on the MERN stack, which includes:

- **React.js:**
 - **Purpose:** A popular JavaScript library for building dynamic and responsive user interfaces.
 - **Why:** React component-based architecture makes it easy to develop reusable UI components, while its virtual DOM improves performance and rendering efficiency.
- **Node.js:**
 - **Purpose:** A JavaScript runtime built on Chrome's V8 JavaScript engine for executing JavaScript server-side.
 - **Why:** Node.js allows for fast, non-blocking, and event-driven I/O, making it ideal for building scalable, real-time web applications like an online store.

3. Database:

- **MongoDB:**
 - **Purpose:** A NoSQL database used for storing and managing data.
 - **Why:** MongoDB's flexible, document-based storage makes it a good fit for dynamic data like user profiles, books, orders, and reviews, and it scales easily for high-volume applications.
- **Express.js:**
 - **Purpose:** A minimal and flexible Node.js web application framework used to build RESTful APIs.
 - **Why:** Express simplifies backend development, routing, and handling HTTP requests, making it easier to structure and manage backend services.

Additional Technologies

JWT (JSON Web Tokens):

- **Purpose:** Used for securing user authentication and authorization.
- **Why:** JWT allows users to log in and securely access protected routes without requiring them to re-authenticate for every request.

Material-UI / Bootstrap :

- **Purpose:** Pre-built UI components for creating responsive, visually appealing, and consistent design.
 - **Why:** Reduces development time and ensures a consistent, mobile-friendly layout.
-

4. System Requirements

Hardware:

- Windows 8 or higher machine with a stable internet connection (30 Mbps recommended).

Software:

- Node.js (latest version)
- MongoDB Community Server
- Two web browsers (for testing purposes)

System Required:

- Bandwidth of 30mbps
-

5. Features

Key Features

1 . User Registration and Authentication:

- Secure user login and registration with **JWT** for authentication and **Bcrypt.js** for password encryption, ensuring data privacy and security.

2 . Search and Browse Books:

- Advanced search functionality and filtering options to help users easily find books by title, author, genre, or price range, enhancing the browsing experience.

3 . Shopping Cart and Checkout:

- Users can add books to a **persistent shopping cart**, modify quantities, and proceed to a secure **checkout** with payment integration via **Stripe** or **PayPal**.

4 .Order History and Tracking:

- Users can view and track the status of their orders, including shipping updates and past purchase details, offering a seamless post-purchase experience.

5 . Admin Dashboard:

- A powerful **admin dashboard** to manage books (add/update/remove), process orders, and monitor sales, inventory, and user activity.

6 . Book Reviews and Ratings:

- Users can **rate and review** books, helping others make informed decisions based on feedback from the community.

7 . Responsive Design:

- The application is **mobile-first** and fully responsive, ensuring a smooth and optimized shopping experience across all devices (desktop, tablet, and mobile).

6. Project Architecture

- **Frontend (React.js):**

The frontend is built with React.js, leveraging a component-based architecture for UI modularity and reusability. React Router handles smooth navigation between different pages like product listings, book details, and the shopping cart. State management is managed globally using Redux, allowing the application to efficiently manage user data, shopping cart, and order information while ensuring a responsive, mobile-first design.

- **Backend (Node.js + Express.js):**

The backend is powered by Node.js with Express.js for routing and API management. The server handles authentication via JWT (JSON Web Tokens) and uses middleware for security and data validation. The backend processes business logic, interacts with the database, and ensures smooth communication with external services like payment gateways (Stripe/PayPal) for secure transactions.

- **Database (MongoDB):**

MongoDB is used as the database for its flexibility and scalability. The data is structured in collections, such as **Users**, **Books**, and **Orders**, allowing for fast reads and writes. Mongoose is used for data modelling, schema validation, and performing CRUD operations. The database supports real-time updates and enables easy horizontal scaling to handle growth in users, products, and transactions.

7. Installation and Setup

Step-by-Step Setup for the Online Book Store App (MERN Stack)

1. Prerequisites

1. Node.js (v14 or higher) and npm (comes with Node.js)
2. MongoDB (local or MongoDB Atlas for cloud)
3. Git (optional for cloning the repository)
4. Text Editor/IDE (e.g., VSCode)

2. Backend Setup (Node.js + Express.js)

1. Navigate to the backend folder.
2. Run `npm install` to install the backend dependencies.

3. Create a .env file and add the necessary environment variables (e.g., MongoDB URI, JWT secret).
4. Run the backend server with npm start.

3. Frontend Setup (React.js)

1. Navigate to the frontend folder.
2. Run npm install to install the frontend dependencies.
3. Create a .env file with the backend API URL.
4. Start the frontend with npm start .

4. Build and Deploy for Production

1. Build the React app using npm run build.
2. Serve the React app with the Express backend (optional).
3. Deploy the frontend to platforms like Netlify and the backend to Heroku or AWS.

5. Troubleshooting

- If you encounter CORS issues, install and configure the cors middleware in the backend.
- Ensure the MongoDB connection URI is correct, and if using Atlas, check IP whitelist settings.

6. Access the App:

- Frontend: http://localhost:3000
- Backend: http://localhost:8000

8. Workflow and Usage

User Roles and Functionalities

1. **Customer:**
 - **Browse Books:** View and search available books.
 - **Add to Cart:** Add books to shopping cart for purchase.
 - **Sign Up/Login:** Create an account or log in to track orders.
 - **Checkout:** Enter payment details and complete the purchase.
 - **Order History:** View past orders and order status.
2. **Admin:**
 - **Login:** Access the admin dashboard with special credentials.
 - **Manage Books:** Add, update, or delete books in the store.
 - **View Orders:** See all customer orders and update their status.
 - **Manage Users:** View and manage customer accounts.
 - **Track Sales:** Monitor overall sales and revenue.

Workflow

1. **Frontend:**

- **User** visits the site, browses books, adds to cart, and checks out.
- **Admin** logs in to manage books, view orders, and track user activity.
- 2. **Backend:**
 - **API** handles requests for book details, user management, and order processing.
 - **Database (MongoDB)** stores book, user, and order data.
- 3. **Authentication:**
 - **JWT** ensures only authorized users (customers and admins) can perform sensitive actions (e.g., order, manage books).

9. Folder Structure

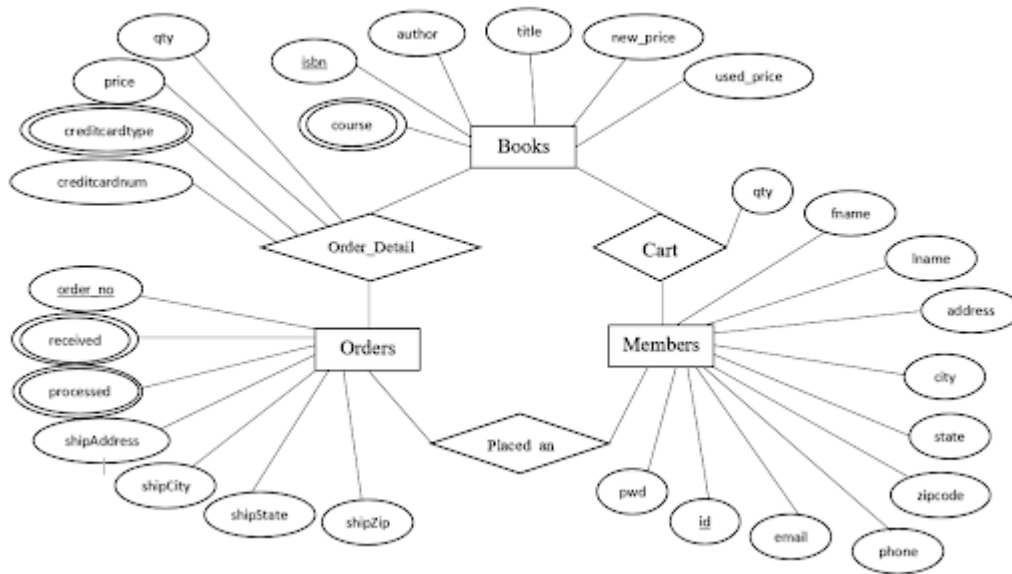
online-book-store/

```

|
|— backend/           # Backend (Node.js/Express)
|   |— controllers/   # Logic for handling requests (e.g., books, users)
|   |— models/        # MongoDB schemas (e.g., books, users, orders)
|   |— routes/        # API routes (e.g., /books, /users, /orders)
|   |— middleware/    # Authentication and error handling
|   |— .env           # Environment variables (e.g., MongoDB URI, JWT secret)
|   |— server.js      # Main server file
|   └─ package.json   # Backend dependencies
|
|— frontend/         # Frontend (React)
|   |— components/    # Reusable UI elements (e.g., Header, Footer, BookItem)
|   |— pages/         # Different pages (e.g., Home, Cart, Checkout)
|   |— services/      # API calls to the backend
|   |— App.js         # Main React component
|   |— index.js       # Entry point for React app
|   └─ package.json   # Frontend dependencies
|
└─ README.md         # Project documentation

```

10. ER Diagram



11.Challenges Faced

- **Authentication and Authorization:** Managing secure login and user roles (customer vs admin).
- **Payment Integration:** Ensuring smooth and secure payment processing for orders.
- **Database Management:** Efficiently organizing and querying large amounts of data (books, orders, users).

12. Future Enhancements

- **Recommendation System:** Implement personalized book recommendations based on user preferences and browsing history.
- **Multi-language Support:** Add support for multiple languages to make the app accessible to a broader audience.
- **Mobile App:** Develop a mobile app version for iOS and Android to reach more users and improve accessibility.

13. Project Implementation & Execution

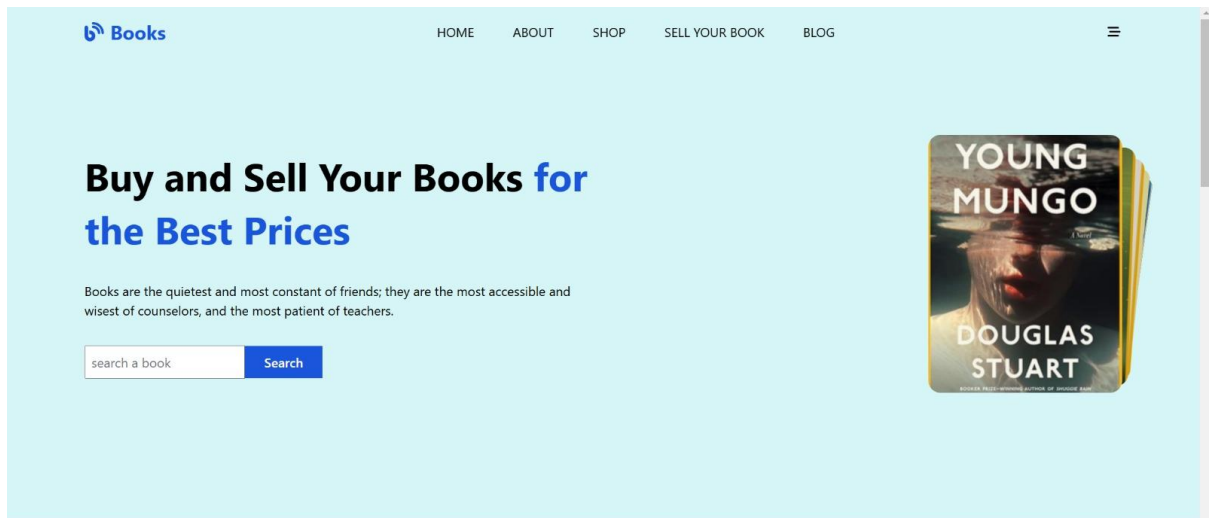
The Online Book Store app is developed using the MERN stack (MongoDB, Express, React, Node.js). It allows users to browse, purchase, and manage books, while admins can add or update book details and process orders. Key features include user authentication, shopping cart management, and secure payment integration. The project involves backend API development, frontend UI/UX design, database management, and deployment to cloud platforms like Heroku and Netlify.



- **LOGIN PAGE**

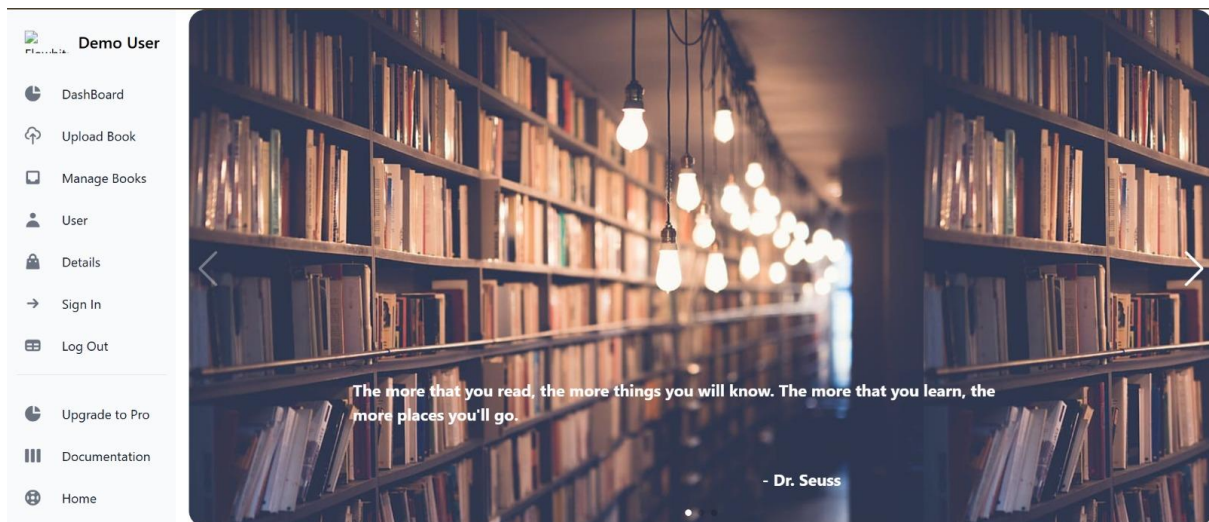
A login form UI mockup. The form is titled "Log in Form" and is set against a background of blue overlapping rectangles. It contains two input fields: the first is for an email address, with "sumanth@gmail.com" entered; the second is for a password, represented by a series of dots. Below the password field is a link that says "If you don't an account. Please [SignUp](#) here". At the bottom of the form is a blue "Log in" button. Below the button is a horizontal line, and then a "Login with Google" button featuring the Google logo.

- **MAIN PAGE:**





Best Seller Book


- **SELLING PAGE:**





- **UPLOAD PAGE:**


 Demo User


 Dashboard


 Upload Book


 Manage Books


 User


 Details

 Sign In

 Log Out

 Upgrade to Pro

 Documentation

 Home

Upload A book

Book Title

Book Name

Author Name

Author Name

Book image URL

Book image URL

Book Catagory

Fiction

Book Description

write you book description...

Book PDF URL

book pdf url

Upload Book

14. Conclusion

The **Online Book Store** app, built with the **MERN stack**, provides a fully functional, scalable platform for browsing and purchasing books, along with admin tools for managing inventory and orders. Key features like secure user authentication, a shopping cart, and integrated payment processing ensure a smooth and reliable user experience. Leveraging MongoDB, Express, React, and Node.js, the app demonstrates an efficient full-stack solution, with potential for future enhancements such as personalized recommendations and mobile app support.