Full Stack Development with MERN

Project Documentation format

1. Introduction

- **Project Title:** Online Bookstore Application
- **Team Members:**

1. Varsha M S – Frontend Developer
2. Roshini S – Backend Developer
3. Durga K – Database Manager
4. Leena Rani S - Documentation
5. Vinitha V – Installation and Implementation

2. Project Overview

- **Purpose:**
  - The "Online Bookstore Application" project aims to provide a comprehensive, user-friendly platform for users to browse, purchase, and manage books online.
  - The primary goal is to streamline the shopping experience for book lovers by offering a virtual bookstore where they can explore a wide range of genres, view book details, read reviews, and make purchases conveniently.
  - For the business, the application also serves as a tool to manage inventory, track orders, and analyze sales.

- **Features:**

  - **User Accounts and Profiles:** Allows users to create accounts, view order history, and manage personal information.
  - **Advanced Search and Filters:** Users can search for books by title, author, genre, or ISBN, with filters for price, publication year, and popularity.
  - **Book Details and Reviews:** Provides detailed information on each book, including author info, synopsis, ratings, and user reviews.
  - **Shopping Cart and Wishlist:** Enables users to add items to the cart for immediate purchase or save them to a wishlist for later.
  - **Secure Checkout and Payment Gateway:** Facilitates secure transactions with multiple payment options, including credit/debit cards, PayPal, and e-wallets.
  - **Order Tracking and Notifications:** Users receive updates on order status, from processing to delivery.
  - **Admin Panel for Inventory and Sales Management:** Admins can add or remove books, track inventory levels, view sales data, and generate reports.

3. Architecture

- **Frontend Architecture (React):**

The frontend of the Online Bookstore Application is built with **React**, creating a dynamic and interactive user interface. Key components and architecture include:

i. **Component-Based Structure:** The UI is divided into reusable components, including Header, Footer, BookList, BookDetails, Cart, and UserProfile. Each component manages its state and can be reused across different views.

ii. **React Router:** Enables seamless navigation between pages, such as Home, Product Listings, Book Details, Cart, and User Profile, without full page reloads.

iii. **State Management (Redux or Context API):** Manages the global state for cart items, user authentication, and product listings to provide a consistent experience across components.

iv. **Responsive Design:** Styled using CSS-in-JS or libraries like Bootstrap or Material-UI to ensure compatibility across devices, providing an optimized experience for both desktop and mobile users.

v. **API Integration with Axios:** Axios is used to communicate with the backend APIs, fetching and updating data for actions like searching for books, adding items to the cart, and updating user profiles.

vi. **Form Validation:** Libraries like Formik and Yup handle user inputs for signup/login, checkout, and reviews, enhancing data accuracy and user experience.

- **Backend Architecture (Node.js and Express.js):**

The backend, built with **Node.js** and **Express.js**, acts as a server to handle client requests, manage business logic, and interact with the database.

i. **RESTful API Endpoints:** Organized with routes to handle requests for books (/books), users (/users), orders (/orders), and reviews (/reviews). Each route uses Express.js for efficient routing and middleware management.

ii. **Controller-Service Pattern:** Controllers handle HTTP requests and responses, while service layers contain the business logic, such as processing payments or calculating discounts.

iii. **Authentication and Authorization:** JSON Web Tokens (JWT) are used for secure user authentication, while role-based access control (admin vs. regular user) restricts certain actions (e.g., only admins can modify inventory).

iv. **Middleware for Security and Validation:** Includes middleware for request validation, error handling, and security headers (using Helmet). Input validation (with libraries like Joi) ensures correct data format.

v. **Payment Processing Integration:** Uses third-party APIs for secure payment processing during checkout.

vi. **Error Handling and Logging:** Structured error handling provides informative responses for both users and developers. Logging with libraries like Winston helps track server activities and troubleshoot issues.

- **Database Schema and Interactions (MongoDB)**

The application uses MongoDB as a NoSQL database to store and manage data.

Schema Design:

- Users Collection: Contains user profiles, including fields for user ID, name, email, hashed password, address, role (user or admin), and wishlist.
- Books Collection: Stores book details, such as book ID, title, author, ISBN, genre, description, price, stock quantity, and average rating.
- Orders Collection: Contains order details like order ID, user ID, order date, ordered books (array with book ID, quantity, and price), total price, and status (processing, shipped, delivered).
- Reviews Collection: Includes book reviews with fields for review ID, user ID, book ID, rating, and review text.

Database Interactions:

- CRUD Operations: MongoDB handles CRUD operations for users, books, orders, and reviews. Mongoose ORM is used to define models and schemas, simplifying interactions with MongoDB.
- Indexes for Faster Queries: Indexes on commonly searched fields (e.g., book title, author, ISBN) ensure efficient searching and filtering.
- Aggregation Pipelines: Used to calculate metrics, such as the average rating for a book or the total sales per genre.
- Data Validation and Schema Enforcement: Mongoose schemas enforce data structure, while MongoDB's flexible schema allows for changes if new features are added.

This architecture offers a robust, scalable solution, ensuring an efficient user experience while simplifying database management and enhancing security.

4. Setup Instructions

- **Prerequisites:**
  - **Node.js** (v14 or higher)
  - **MongoDB** (either local installation or MongoDB Atlas for cloud-based database)

- **Git** (for version control and cloning)
- Optional: **Postman** (for API testing)

- **Installation:**

i. **Clone the Repository:**

```
git clone https://github.com/username/online-bookstore.git
cd online-bookstore
```

ii. **Install Dependencies:**

- Frontend:
  cd client
  npm install

- Backend:
  cd ../server
  npm install

iii. **Set Up Environment Variables:**

- Create .env files in both client and server directories with the following variables:

- **Frontend (client/.env):**

```
REACT_APP_API_URL=http://localhost:5000/api
```

- **Backend (server/.env):**

```
MONGODB_URI=your_mongo_db_connection_string
JWT_SECRET=your_jwt_secret
PORT=5000
```

5. Folder Structure:

✓ **Client (React Frontend):**

```
client/
├── public/          # Public assets and index.html
├── src/
│   ├── components/   # Reusable components (Header, Footer, BookCard, etc.)
│   ├── pages/        # Page components (Home, BookDetails, Cart, etc.)
│   ├── redux/        # State management (slices, reducers, store configuration)
│   ├── services/     # API requests using Axios
│   ├── App.js        # Main application component
│   └── index.js      # Entry point
└── package.json
```

- ✓ **Server (Node.js Backend):**

```
server/
├── config/          # Config files for MongoDB and environment setup
├── controllers/     # Controller functions for handling API requests
├── middleware/      # Authentication, error handling middleware
├── models/          # Mongoose schemas (User, Book, Order, Review)
├── routes/          # API routes (auth, books, orders, users)
├── utils/           # Helper functions
├── server.js        # Entry point
└── package.json
```

6. Running the Application:

- ✓ **Frontend:**
  cd client
  npm start

- ✓ **Backend:**
  cd server
  npm start

  Both servers should now be running locally, with the frontend accessible at http://localhost:3000 and the backend API at http://localhost:5000.

7. API Documentation:

- ✓ **Books**

- **GET** /api/books - Retrieve a list of all books.
- **GET** /api/books/:id - Get details of a single book.

- ✓ **Users**

- **POST** /api/users/register - Register a new user.
- **POST** /api/users/login - User login.

- ✓ **Orders**

- **POST** /api/orders - Create a new order.
- **GET** /api/orders/:id - Retrieve order details.

- ✓ **Request Example:**

```
POST /api/users/login
{
  "email": "example@domain.com",
  "password": "password123"}
```

✓ **Response Example:**

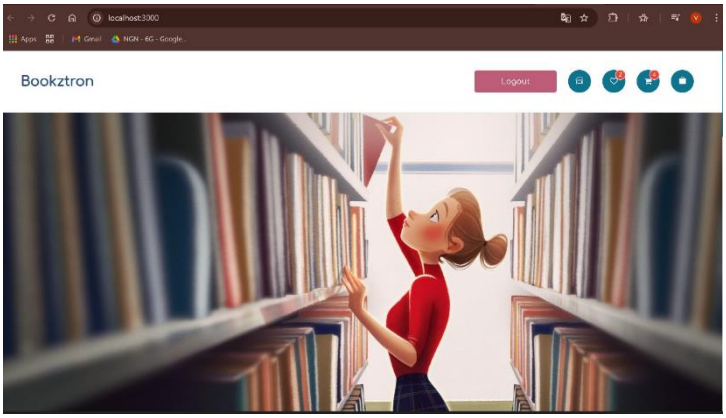
```
{
      "token": "eyJhbGciOiJIUzI1NiIsInR...",
      "user": {
        "id": "6123abc456def789",
        "name": "John Doe",
      "email": "example@domain.com"
      }
 }
```

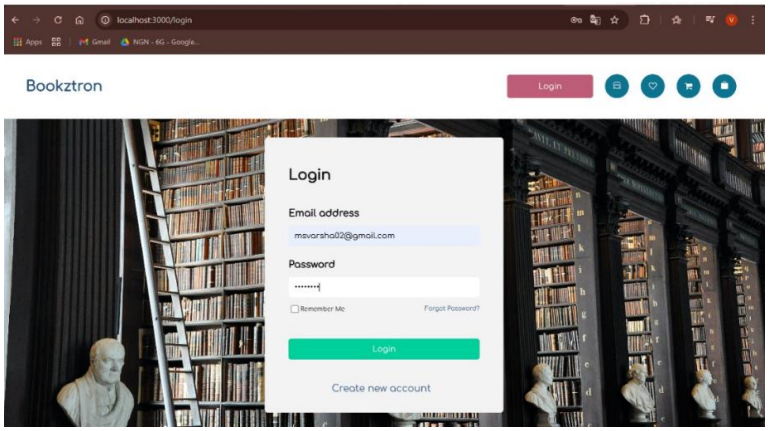
8. Authentication:

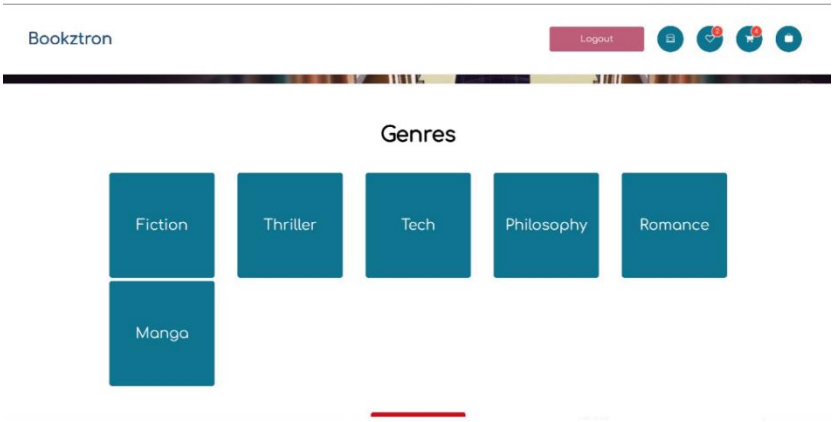
- **Authentication:** JWTs are used for secure user authentication. After logging in, the server generates a token that the client must include in the headers for subsequent requests.

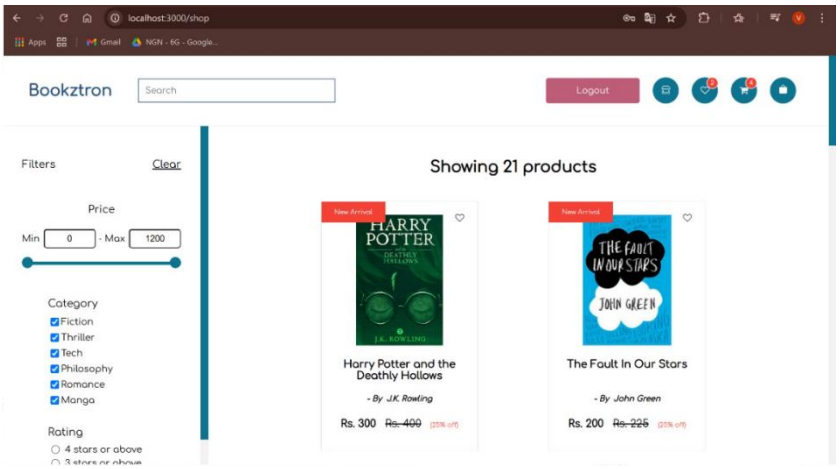
- **Authorization:** Role-based access control ensures only admins can perform actions like adding or deleting books.

9. User Interface:






Bookztron
Logout
Genres
Fiction
Thriller
Tech
Philosophy
Romance
Manga


localhost:3000/shop
Bookztron
Search
Logout
Filters
Clear
Price
Min 0 - Max 1200
Category
Fiction
Thriller
Tech
Philosophy
Romance
Manga
Rating
4 stars or above
Showing 21 products
Harry Potter and the Deathly Hollows
- By J.K. Rowling
Rs. 300 Rs. 400
The Fault In Our Stars
- By John Green
Rs. 200 Rs. 225

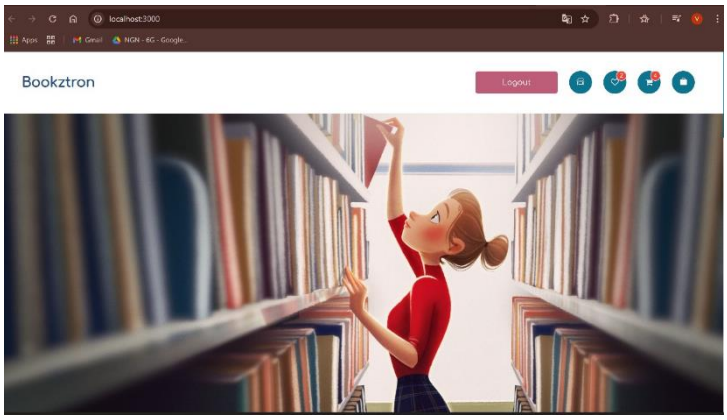

localhost:3000/shop/6230e5bb16d6b222d998c64b
Bookztron
Logout
HARRY POTTER
DEATHLY HALLOWS
J.K. ROWLING
Harry Potter and the Deathly Hollows
Author : J.K. Rowling
Description : Harry Potter and the Deathly Hallows is a fantasy novel written by British author J. K. Rowling and the seventh and final novel of the main Harry Potter series. It was released on 14 July 2007 in the United Kingdom by Bloomsbury Publishing, in the United States by Scholastic, and in Canada by Raincoast Books.
Rating : 4
Rs. 300 Rs. 400 (25% off)
Add to wishlist
Add to cart

10. Testing:

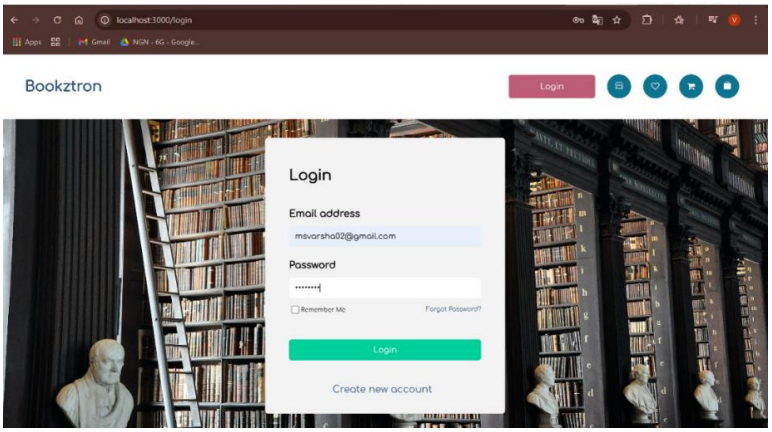
- **Frontend Testing:** Using Jest and React Testing Library for component testing and UI validation.
- **Backend Testing**: Using Mocha and Chai for unit tests of routes and controllers.
- **End-to-End Testing:** Tools like Cypress simulate user actions across the entire app.

11. Screenshots or Demo

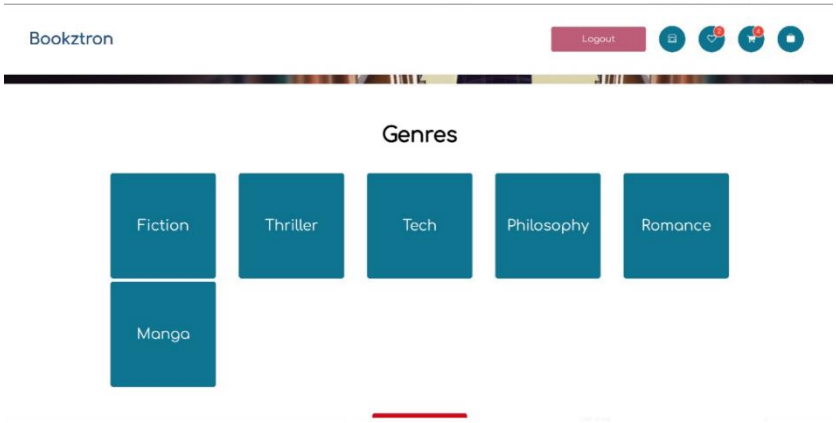
Front Page:



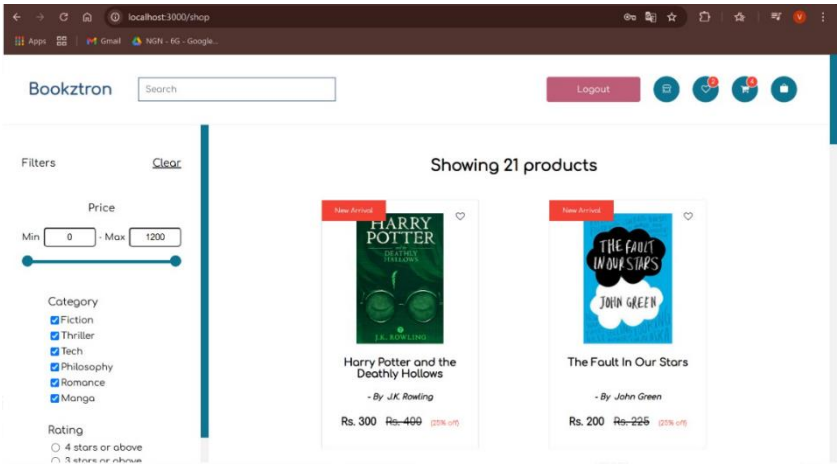
Login Page:



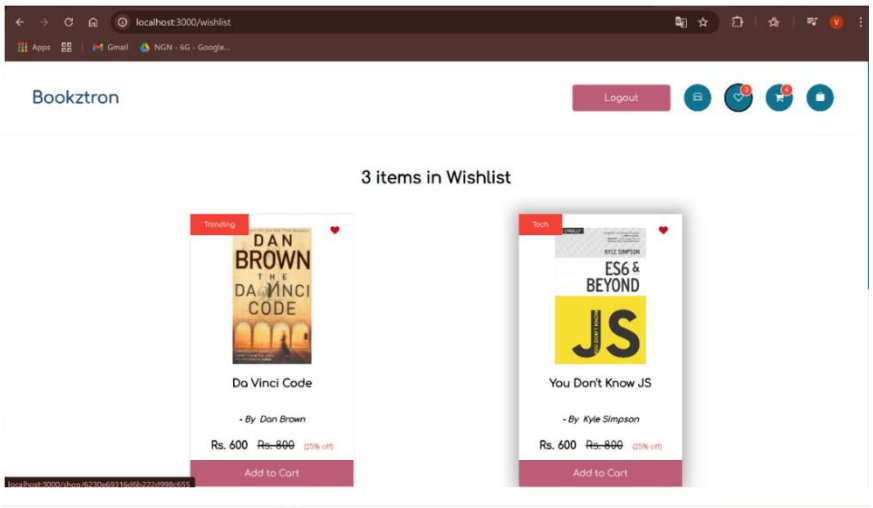
Menu:



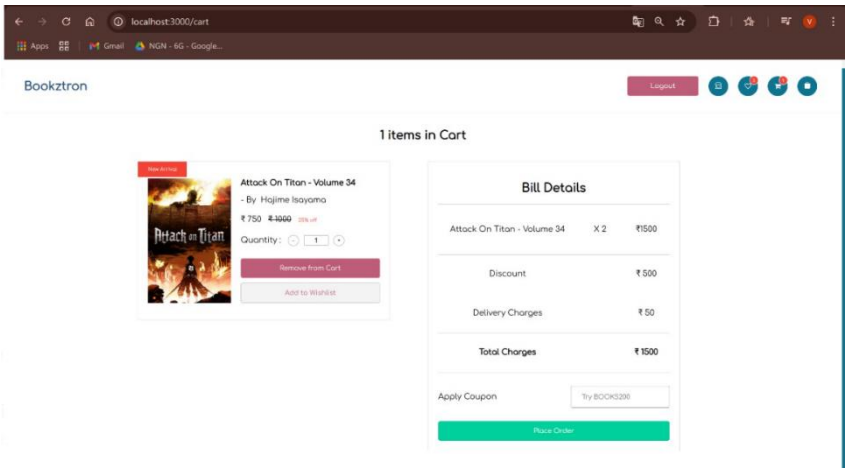
Product display:



Wishlist:



Cart:



12. Known Issues:

Occasionally, certain items may not sync with the cart due to network latency. Plans are underway to optimize the API handling for better response time.

13. Future Enhancements:

- **Wishlist Sharing:** Allow users to share wishlists with friends or on social media.
- **Recommendation Engine**: Recommend books to users based on previous purchases.
- **Discount System**: Admins can apply discounts or special offers, and users can redeem coupons during checkout.