**BOOK STORE - ONLINE BOOK STORE PROJECT**

**1.Introduction**

**Project Title:** BOOK STORE – ONLINE BOOK STORE

**TEAM MEMBERS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO** | **NAME** | **ROLE** | **RESPONSIBILITIES** |
| 1 | **R. Shreya Janaki** | Full-Stack  Developer | Responsible for  Overall development, including front-end, back-end server-side logic, and database design |
| 2 | **G.Sharen Roseline** | Frontend Developer | Responsible for UI/UX design using react material UI and bootstrap |
| 3 | **R.Sherlin Swetha** | Database Administrator | Responsible for MongoDB setup and ensuing data integrity |
| 4 | **R.Shankar Madhava** | Backend Developer | Responsible for express.js setup and API development |

**2.Project Overview:**

Welcome to the Book Store project, a fully functional online bookstore built using the MERN (MongoDB, Express.js, React.js, Node.js) stack. This project allows users to browse, add, edit, and delete book entries, providing an intuitive and efficient way to manage your book collection.

**Features**

**Browse Books**

* Users can easily browse through the list of books, either in table format or by specific card view, providing flexibility in how they view their collection.

**Add New Books**

* Add new entries for books to keep your collection up to date. Fill in the details, such as title, author, genre, and more, to maintain an organized library.

**Edit Book Details**

* Users have the ability to edit the details of existing books, ensuring that the information in the collection is accurate and up-to-date.

**Delete Books**

* Remove books from the list when they are no longer in your collection or for any other reason, ensuring that your library stays well-maintained.

**Show Single Book**

* To see single book information.

**3. Tech Stack**

This project is built using the following technologies:

* **MongoDB**: A NoSQL database for storing book information.
* **Express.js**: A Node.js web application framework for building the server-side API.
* **React.js**: A JavaScript library for building the user interface and frontend components.
* **Node.js**: A JavaScript runtime environment for running server-side code.
* **Tailwind CSS**: A utility-first CSS framework for styling the frontend.
* **Axios**: A promise-based HTTP client for making API requests.

**4. Folder Structure**

The project has the following folder structure:

Book Store/

├── backend/

├── frontend/

├── README.md

* frontend/: Contains all the frontend code.
  + frontend/src/: Includes all frontend components and page routes.
* backend/: Houses all the database models and backend logic.
  + backend/routes/: Contains all the route handlers.

**5. How to run this project:**

**For Frontend**

Follow the below steps to run the project:

* First clone or unzip the project folder.
* Go to the frontend directory by using the following command cd frontend.

>>> Stepup firebase app and configure the environment

VITE\_API\_KEY="AIzaSyB-6Mt2LZrRaPgXeNei15BoGBdRmSworD8"

VITE\_Auth\_Domain="book-store-d59c4.firebaseapp.com"

VITE\_PROJECT\_ID="book-store-d59c4"

VITE\_STORAGE\_BUCKET="book-store-d59c4.firebasestorage.app"

VITE\_MESSAGING\_SENDERID= "1003381657538"

VITE\_APPID="1:1003381657538:web:03cb93ffbbe9e1553b905b"

* Then run npm install commend to install node dependencies.
* Finally, to run the project, use npm run dev command.

**For Backend**

Follow the below steps to run the project:

* First clone or unzip the project folder.
* Go to the backend directory by using the following command  cd backend.

DB\_URL=**"mongodb+srv://mohith:mohith@book-store.ikkkt.mongodb.net/ebooks-collection?retryWrites=true&w=majority&appName=Book-Store"**

JWT\_SECRET\_KEY=**“bc992a20cb6706f741433686be814e3df45e57ea1c2fc85f9dbb0ef7df12308a669bfa7c976368ff32e32f6541480ce9ec1b122242f9b1257ab669026aeaf16”**

* Then run npm install commend to install node dependencies.
* Finally, to run the project, use npm run start:dev command.

6. Setup Instructions

Prerequisites:

* + Node.js: **Make sure Node.js is installed. This is needed to run the application locally and install necessary dependencies.**
  + MongoDB: **Ensure that you have a running instance of MongoDB. You can use either a local MongoDB setup or a cloud-based solution like MongoDB Atlas.**
  + npm: **The Node Package Manager (npm) is required to install dependencies.**

Installation:

1. **Clone the Repository:**

**git clone https://github.com/your-repo/book-a-doctor.git**

1. **Install Frontend Dependencies: Navigate to the client folder and run:**

**cd client**

**npm install**

1. **Install Backend Dependencies: Navigate to the server folder and run:**

**cd server**

**npm install**

1. **Set up Environment Variables: In the server folder, create a .env file to store sensitive information such as:**

**MONGODB\_URI: The connection string for your MongoDB database.**

**JWT\_SECRET: A secret key used for signing JWT tokens.**

**PORT: Port on which the backend server will run (default is 5000).**

1. **Start the Backend Server: Run the following command in the server directory:**

**npm start**

1. **Start the Frontend Server: Run the following command in the client directory:**

**npm start**

**The application will be available at http://localhost:3000 (frontend) and http://localhost:5000 (backend).**

**7. Running the Application**

Frontend: **To start the frontend server, run the following command in the client folder:**

**npm start**

Backend: **To start the backend server, run the following command in the server folder:**

**npm start**

8. Authentication

JWT Authentication: **In this system, users (patients and doctors) authenticate by logging in and receiving a** JWT (JSON Web Token). **This token must be included in the authorization header of all future requests to protected routes. The token serves as proof that the user is authenticated and ensures that only authorized users can access specific resources, like booking appointments or accessing the doctor’s dashboard.**

Authorization Middleware:

* Authentication Middleware: **This middleware verifies whether a user is logged in by checking the validity of the JWT token. If the token is valid, the user is allowed to proceed; otherwise, access is denied.**
* Authorization Middleware: **Once the user’s identity is verified through authentication, this middleware checks if the user has the appropriate role to access a certain route. For example, only doctors should be able to access doctor-specific routes (e.g., managing their schedule), while patients should be restricted to booking appointments and viewing their own data.**

9. User Interface

The user interface (UI) for an online bookstore using the MERN stack involves designing and developing the frontend (React) and connecting it to the backend (Node.js and Express) with MongoDB as the database. Here’s a structured approach to creating the UI:

**Homepage:**

* + Display featured books, categories, and new arrivals.
  + Search bar for books by title, author, or genre.

**Book Listings Page:**

* + Show books as cards or a list with thumbnails, titles, authors, and prices.
  + Filters for genre, price range, rating, and more.
  + Pagination or infinite scroll.

**Book Details Page:**

* + Detailed view of the book, including:
    - Title, author, genre, price.
    - Description and reviews.
    - Add to cart/wishlist button.

**Shopping Cart:**

* + View selected books with their quantities and prices.
  + Update quantities or remove items.
  + Proceed to checkout button.

**User Authentication:**

* + Login and signup pages.
  + Integration with OAuth providers (e.g., Google, Facebook).

**User Profile:**

* + View order history and manage personal details.
  + Update passwords.

**Admin Panel (optional):**

* + Manage books (add/edit/delete).
  + View orders and user data.

**Checkout Flow:**

* + Billing and shipping information.
  + Payment gateway integration (e.g., Stripe, Razorpay).

10. Testing

Testing Strategy:

Unit Testing:

* + - **Used for testing individual components and backend functions.**
    - **Jest and Mocha were used for unit testing backend routes, validation, and middleware.**

Integration Testing:

* **Focused on ensuring that the communication between the frontend, backend, and database worked as expected.**
* **Tools like Postman were used to test API endpoints.**

End-to-End Testing:

* **Simulated the user journey from login to booking an appointment and managing it.**
* **Cypress was used for front-end testing, ensuring that all user interactions are properly handled.**

Testing Tools:

Frontend: **Jest, Enzyme, React Testing Library for unit and integration testing of React**

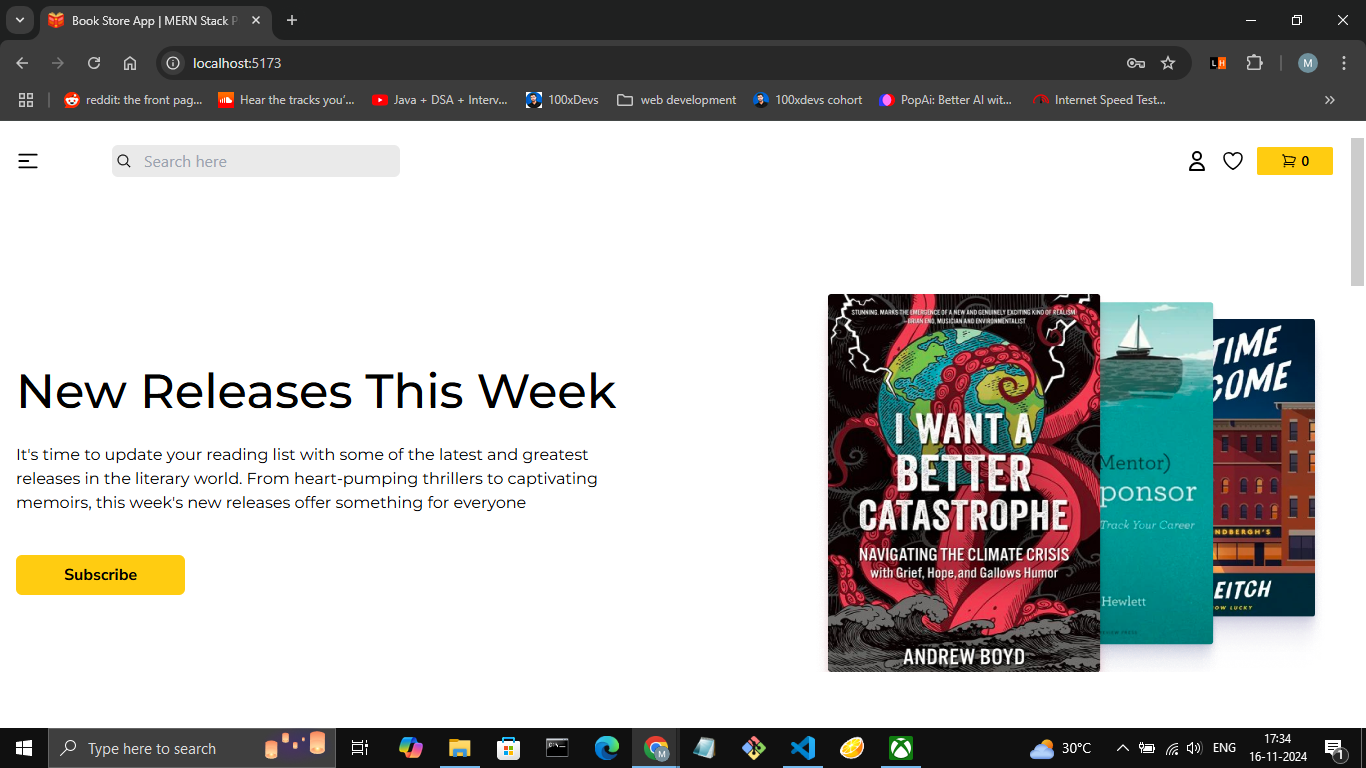
**components.**

Backend: **Mocha, Chai, Supertest for testing Express.js API routes.**

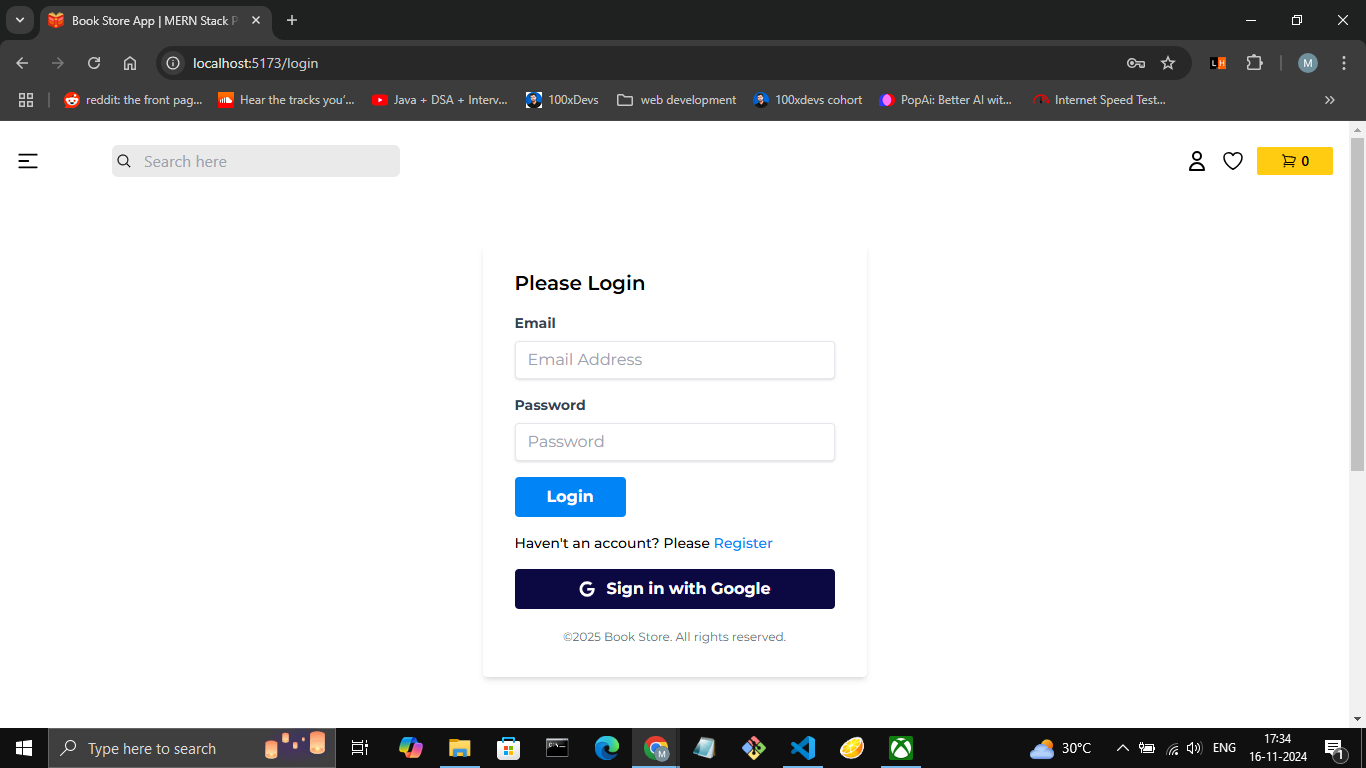
API Testing: **Postman for manually testing backend endpoints.**

**11. Screenshots**

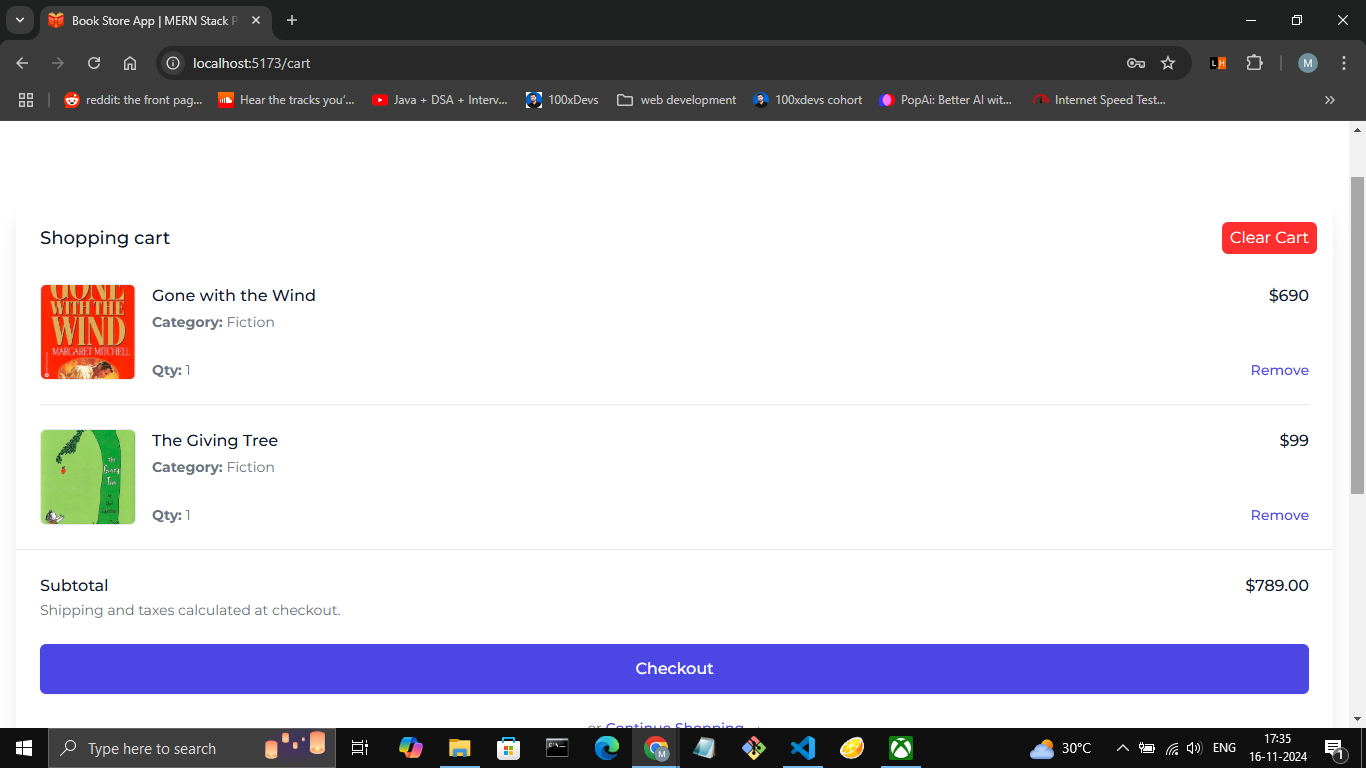
Landing page:



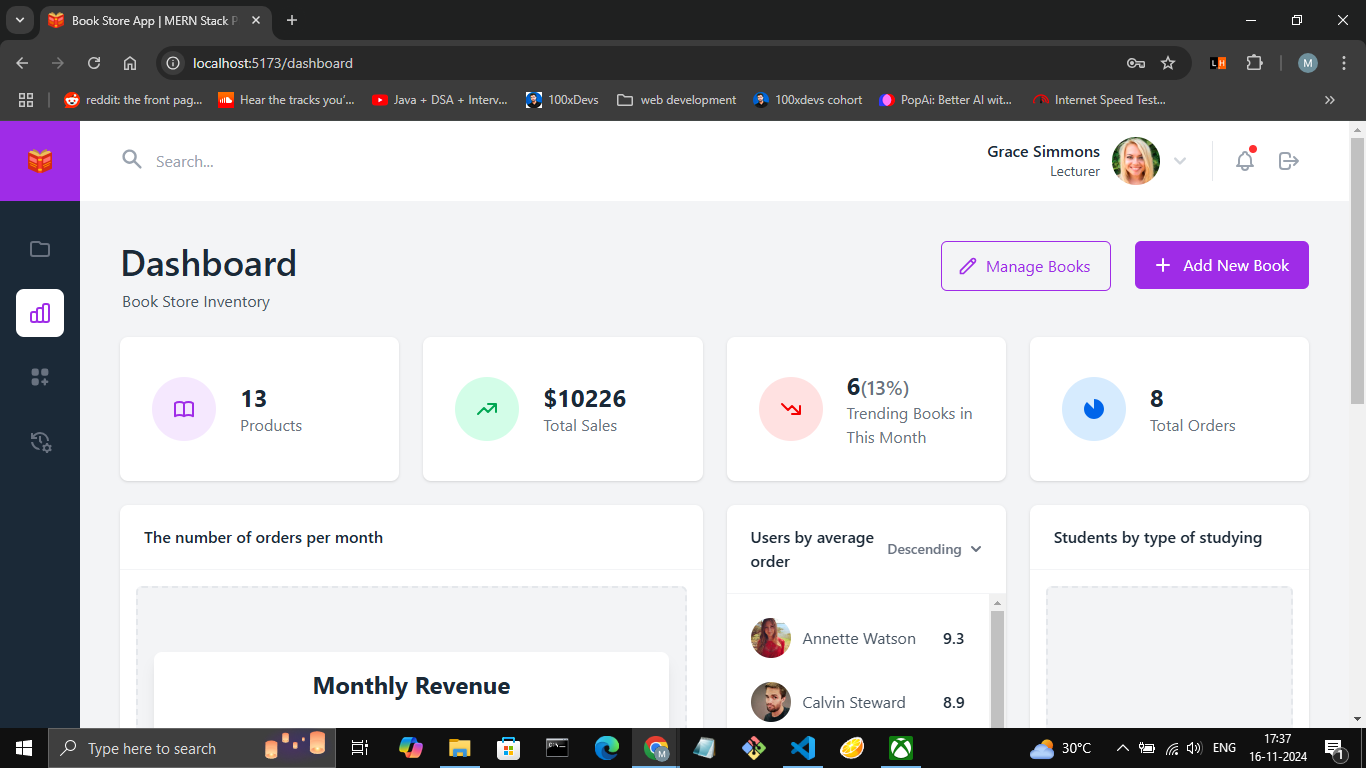
Login Page:



**Cart Page:**

****

**User Dashboard:**

****

12. Known Issues

**While most of the application is fully functional, there are a few issues that developers or users should be aware of:**

Issue with Appointment Confirmation:

* **Occasionally, the appointment confirmation status may take a few moments to update due to delays in backend processing.**

Responsive Layout on Mobile:

* **Some UI components may not fully adjust for extremely small screen sizes (e.g., older smartphones).**

File Upload Issue:

* **The file upload feature for appointment documents may fail intermittently with larger files (greater than 10MB).**

Workarounds:

* **For appointment confirmation delays, refreshing the page can help load the updated status.**
* **Mobile users can zoom out to view the full content if it's cut off on smaller screens.**
* **For file uploads, users are encouraged to compress files before uploading.**

13. Future Enhancements

**1. Advanced Search and Recommendations**

* **Enhancements:**
  + Implement full-text search using MongoDB's text indexes.
  + Add filters like genre, author, price range, and rating.
  + Use **collaborative filtering** or **content-based filtering** for book recommendations.
* **Technology:**
* Use **Elasticsearch** for advanced search capabilities.
* Implement a **Recommendation Engine** using machine learning (e.g., TensorFlow.js or Python Flask API).

**2. Personalized User Experience**

* **Enhancements:**
  + Show personalized book recommendations based on browsing or purchase history.
  + Allow users to create wishlists.
  + Offer a personalized dashboard for users to track orders, favorite authors, or genres.
* **Technology:**
  + Use **Redux** or **Context API** for maintaining user preferences.
  + Store personalized data in MongoDB collections.

**3. Mobile App Integration**

* **Enhancements:**
  + Create a mobile-friendly version of the website.
  + Develop a mobile app using **React Native** to provide a seamless shopping experience.
* **Technology:**
  + Use a **REST API** or **GraphQL API** to connect the app with the backend.

**4. Social Features**

* **Enhancements:**
  + Allow users to review and rate books.
  + Add discussion forums for book lovers to connect.
  + Enable users to share book links on social media platforms.
* **Technology:**
  + Integrate **Auth** for login (Google, Facebook).
  + Use **Disqus API** for discussion forums or create a custom forum.

**5. AI-Powered Virtual Assistant**

* **Enhancements:**
  + Add a chatbot for book recommendations, order tracking, and customer support.
  + Enable voice-based search and commands.
* **Technology:**
  + Use **Dialogflow** or **OpenAI GPT-based chatbots**.
  + Integrate **Groq Whisper** for voice-to-text search.

**6. Subscription Model**

* **Enhancements:**
  + Offer a subscription for unlimited book access (e.g., Kindle Unlimited-style).
  + Provide discounts for loyal customers or frequent buyers.
* **Technology:**
  + Use **Stripe** or **Razorpay** to manage recurring payments.