MERN STACK POWERED BY MONGODB

BOOK STORE

A PROJECT REPORT

Submitted by

MITHUN KUMAR D L **au113321104059**
SURYOTHAYAN A **au113321104100**
VISHVAKUMAR L **au113321104116**
GOWTHAMCHANDRAN J **au113321104301**

BACHELOR OF ENGINEERING

COMPUTER SCIENCE AND ENGINEERING

VELAMMAL INSTITUTE OF TECHNOLOGY

CHENNAI 601 204

ANNA UNIVERSITY CHENNAI: 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**BOOK STORE**" is the Bonafide work of "**MITHUN KUMAR D L – 113321104059, SURYOTHAYAN A– 113321104100, VISHVAKUMAR L – 113321104116, GOWTHAMCHANDRAN J – 113321104301**" who carried out the project work under my supervision.

SIGNATURE
Dr.V.P.Gladis Pushparathi,
Professor,
Head Of the Department,
Computer Science And Engineering,
Velammal Institute Of Technology,
Velammal Knowledge Park,
Panchetti, Chennai-601 204

SIGNATURE
Mrs. Joice Ruby J
Assistant Professor,
NM Coordinator,
Computer Science And Engineering,
Velammal Institute Of Technology,
Velammal Knowledge Park,
Panchetti, Chennai – 601 204.

ACKNOWLEDGEMENT

We are personally indebted to many who had helped me during the course of this project work. My deepest gratitude to the God Almighty.

We are greatly and profoundly thankful to our beloved Chairman **Thiru.M.V.Muthuramalingam** for facilitating us with this opportunity. My sincere thanks to our respected Director **Thiru.M.V.M Sasi Kumar** for his consent to take up this project work and make it great success.

We are also thankful to our Advisor **Shri.K.Razak**, **M.Vaasu** and our Principal **Dr.N.Balaji** and our Vice Principal **Dr.S.Soundararajan** for their never ending encouragement which accelerates us towards innovation.

We are extremely thankful to our Head of the Department **Dr.V.P.Gladis Pushaparathi**, Internship coordinator **Mrs. Joice Ruby J** for their valuable teachings and suggestions.

The Acknowledgement would be incomplete if we would not mention a word of thanks to my Parents, Teaching and Non-Teaching Staffs, Administrative Staffs, Friends who had motivated and lent their support throughout the project.

Finally, we thank all those who directly or indirectly contributed to the successful completion of this project .Your contributions have been a vital part of our success

TABLE OF CONTENTS

S.NO.	TITLE	PAGE NO
1	INTRODUCTION	5
2	PROJECT OVERVIEW	6
3	ARCHITECTURE	8
4	SETUP INSTRUCTIONS	11
5	RUNNING THE APPLICATION	13
6	API DOCUMENTATION	15
7	AUTHENTICATION	25
8	TESTING	28
9	SCREENSHOTS	32
10	KNOWN ISSUES	36
11	FUTURE ENHANCEMENTS	39

INTRODUCTION

The Book App Store is an advanced web application designed to revolutionize the way people explore, purchase, and review books online. Built with the MERN stack, which comprises MongoDB, Express.js, React.js, and Node.js, the application leverages the full power of these technologies to deliver a seamless, efficient, and user-friendly experience. The app caters to a diverse range of users, including book enthusiasts, sellers, and administrators, providing them with tailored features that meet their specific needs. At its core, the Book App Store offers a dynamic and comprehensive platform for managing books, integrating secure user authentication, an intuitive interface, and robust backend operations to ensure a seamless user journey.

The Book App Store offers a host of features that make it a standout solution for book commerce. The platform boasts a comprehensive book catalog, where users can browse through an extensive collection of books enriched with detailed information such as titles, authors, genres, prices, and reviews. Advanced search and filter options enable users to quickly find books that match their preferences, enhancing the overall browsing experience. The shopping cart and checkout process are designed to be user-friendly and efficient. Customers can add, update, or remove items from their cart, and the platform dynamically updates the total cost and quantity of selected items.

The seamless integration of the MERN stack enhances the Book App Store's workflow and performance. The frontend, powered by React.js, communicates with the backend using RESTful APIs to fetch and manipulate data. Features like real-time search updates, cart management, and responsive design ensure that users enjoy a smooth and intuitive experience across devices. On the backend, Express.js handles API routing and processes requests efficiently, while middleware ensures that only authenticated and authorized users can perform specific actions. MongoDB serves as the application’s database, storing data in collections that are optimized for fast retrieval and scalability. Node.js ensures that server-side operations are fast and non-blocking, making the platform capable of handling a large number of simultaneous users without compromising performance.

PROJECT OVERVIEW

Welcome to the literary haven of the digital age—introducing our revolutionary Book-Store Application, a masterpiece crafted with precision using the powerful MERN (MongoDB, Express.js, React, Node.js) Stack. Immerse yourself in a world where the love for reading converges seamlessly with cutting-edge technology, redefining the way bibliophiles explore, discover, and indulge in their literary pursuits.

Tailored for the modern book enthusiast, our MERN-based Book-Store Application seamlessly blends robust functionality with an intuitive user interface. From the joy of discovering new releases to the nostalgia of revisiting timeless classics, our platform promises an immersive reading experience customized to cater to your literary preferences.

Fueling the backbone of our application is MongoDB, ensuring a scalable and efficient database infrastructure that facilitates swift access to an extensive collection of literary works. Express.js, with its streamlined web application framework, establishes a responsive and efficient server, while Node.js ensures high-performance, non-blocking I/O operations—resulting in a seamless and enjoyable user experience.

At the heart of our Book-Store Application lies React, a dynamic and feature-rich JavaScript library. Dive into a visually enchanting and interactive interface where every click, search, and book selection feels like a literary journey. Whether you're exploring on a desktop, tablet, or smartphone, our responsive design ensures a consistent and delightful experience across all devices.

Say farewell to the constraints of traditional bookstores and embrace a new era of possibilities with our MERN Stack Book-Store Application. Join us as we transform how you connect with literature, making the discovery of your next favorite read an effortless and enriching experience. Get ready to turn the digital pages of a new chapter in reading, where every book is just a click away, and the literary world is at your fingertips. It's time to open the door to a future where the love for books meets the convenience of modern technology.

FEATURES

1.User Registration and Authentication:

- Allow users to register accounts securely, log in, and authenticate their identity to access the book store platform.

2.Book Listings:

- Display a comprehensive list of available books with details such as title, author, genre, description, price, and availability status.

3.Book Selection:

- Provide users with options to select their preferred books based on factors like genre, author, ratings, and popularity.

4.Purchase Process:

- Allow users to add books to their cart, specify quantities, and complete purchases securely. Upon successful completion, an order is generated, and the inventory is updated accordingly.

5.Order Confirmation:

- Provide users with a confirmation page or notification containing details of their order, including book information, total price, and order ID.

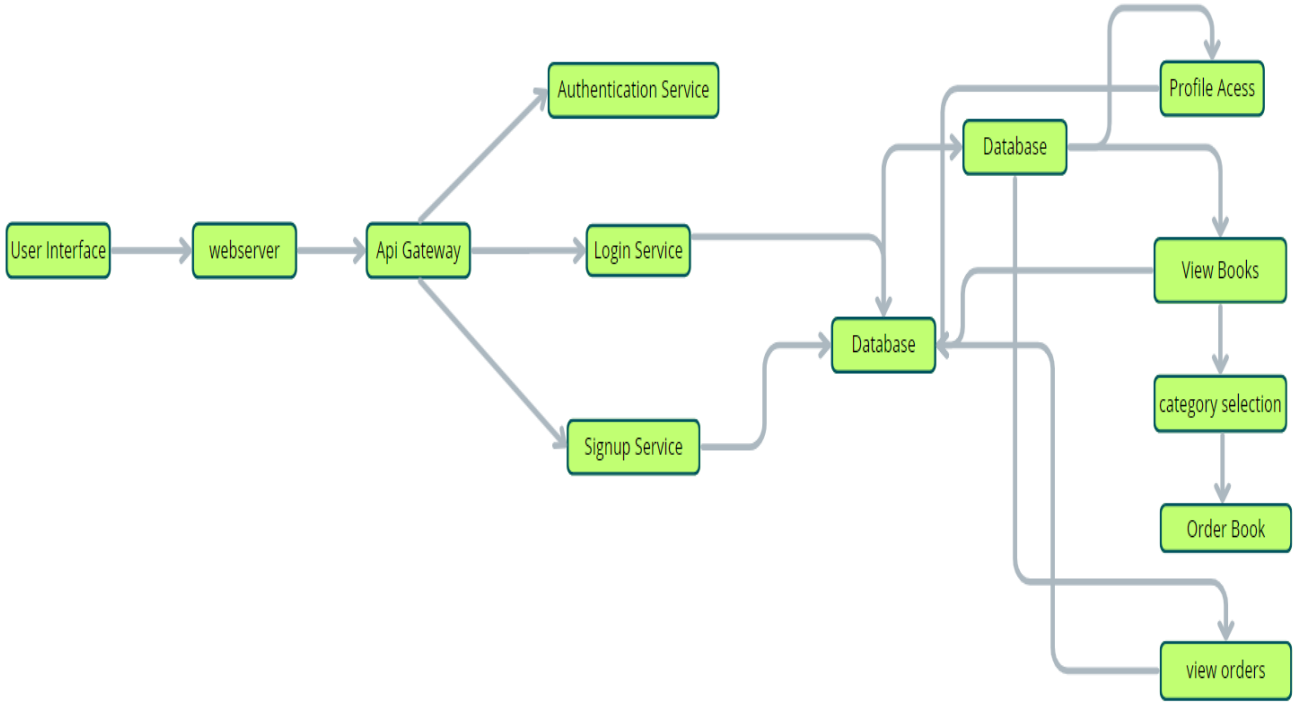
6.Order History:

- Allow users to view their past and current orders, providing options to track shipments, review purchased books, and rate their shopping experience.

This project demonstrates the full capabilities of the MERN stack, combining a robust backend with a responsive and interactive frontend to create a scalable e-commerce platform tailored for book lovers.

ARCHITECTURE

The architecture of the Book Store is designed to be scalable, efficient, and user friendly. It follows a MERN (MongoDB, Express.js, React.js, Node.js) stack to create a full stack application with a focus on modularity and performance. The architecture is divided into three main components: Frontend, Backend, and Database, with seamless communication between them.



COMPONENTS:

1.User Interface:

- The user interface will serve as the platform where customers can browse books, search for specific titles or authors, read book descriptions, and make purchases. It should be intuitive and user-friendly, enabling easy navigation and exploration of available books.

2.Web Server:

- The web server hosts the user interface of the book store app, serving dynamic web pages to users and ensuring a seamless browsing and shopping experience.

3.API Gateway:

- Similar to the original architecture, the API gateway will serve as the central entry point for client requests, directing them to the relevant services within the system. It will handle requests such as fetching book information, processing orders, and managing user accounts.

4.Authentication Service:

- The authentication service manages user authentication and authorization, ensuring secure access to the book store app and protecting sensitive user information during the browsing and purchasing process.

5.Database:

- The database stores persistent data related to books, including information such as titles, authors, genres, descriptions, prices, and availability. It also stores user profiles, purchase history, and other essential entities crucial to the book store app.

6.View Books:

- This feature allows users to browse through the available books. They can explore different categories and genres to discover books of interest.

7.Category Selection:

- Users can select specific categories or genres to filter and refine their book browsing experience, making it easier to find books tailored to their preferences.

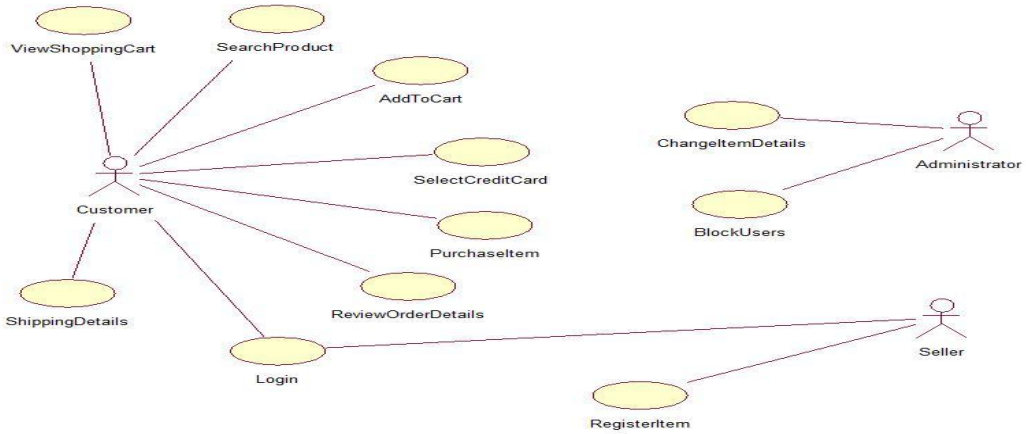
8.Inventory Management Service:

- This service manages information about available books, including their availability, stock levels, and ratings. It ensures efficient management of the book inventory and seamless integration with the browsing and purchasing process.

9.Order Management Service:

- This service facilitates the ordering process, allowing users to add books to their cart, specify quantities, and complete purchases securely. It also handles order tracking and status updates in real-time.

ER Diagram:



User-Book Relationship:

- Type: Many-to-Many (M:M). A single user can read or interact with many books, and a single book can be accessed by many users.
- Implementation: Introduce an intermediate entity, "Interaction", with foreign keys to both User and Book tables. This table could store additional information like reading progress, reviews, or ratings.

Book-Inventory Relationship:

- Type: One-to-Many (1:M). Each book can have multiple copies in inventory, but each copy belongs to one book.
- Implementation: Maintain a separate Inventory table with fields like BookID (foreign key), quantity, location, and condition.

User-Order Relationship:

- Type: One-to-Many (1:M). A single user can place multiple orders, but each order belongs to one user.
- Implementation: Keep the UserID foreign key in the Order table to track user purchase history.

SETUP INSTRUCTIONS

The following detailed setup instructions will guide you through the process of installing and running the Book App Store locally on your system.

PREREQUISITIES

To develop a full-stack Book Store App using React js, Node.js,Express js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

1.Node.js and npm:

Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download: https://nodejs.org/en/download/

• Installation instructions:https://nodejs.org/en/download/package-manager/

2. MongoDB:

Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download: https://www.mongodb.com/try/download/community

• Installation instructions:https://docs.mongodb.com/manual/installation/

3.Express.js:

Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

• Installation: Open your command prompt or terminal and run the following command: **npm install express**

4. Git:

- Download : https://git-scm.com/downloads**.**

5.Browser:

- A modern browser like Google Chrome or Mozilla Firefox.

6.Cloning the Repository:

1. Open a terminal and navigate to the directory where you want to clone the project.
2. Clone the repository.
3. Navigate to the project directory.

7.Environment Variables:

Client:

```
const port = process.env.PORT || 5000
```

Server:

```
PORT=5000
const uri =
"mongodb+srv://gowthamchandranj:mongodbpass@cluster0.4lckv.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";
```

INSTALLATION:

Install the required dependencies for both the client (frontend) and server (backend).

1.Frontend:

Navigate to the client directory:

```
cd my-app-client
```

Install dependencies using npm:

```
npm install
```

2.Backend:

Navigate to the server directory:

```
cd my-app
```

Install dependencies using npm:

```
npm install
```

RUNNING THE APPLICATION

1.Start the MongoDB Service:

- Ensure the MongoDB service is running before starting the application.

```
mongod
```

2.Start the Backend Server:

Navigate to the server directory:

```
cd my-app
```

Start the backend server:

```
npm start
```

or

```
npm run dev
```

The backend server will run at http://localhost:5173/.

3.Start the Frontend Server:

Open a new terminal and navigate to the client directory:

```
cd my-app-client
```

Start the frontend development server:

```
npm start
```

Verify the Setup

1. Open a browser and navigate to http://localhost:5173/ to access the Book App Store
2. Test key functionalities:
   - User registration and login.
   - Books browsing and adding books to the cart.
   - Any functionalities if applicable .

ADDITIONAL NOTES :

Data Seeding:

If the application requires initial data for products or users, run a seed script (if provided) in the backend directory:

npm run seed

Using External APIs:

- If the project integrates external APIs (e.g., payment gateways), ensure their keys are added to the file.

Debugging:

- Use the terminal to monitor logs from the frontend and backend for troubleshooting errors.
- Use browser developer tools for inspecting frontend issues

API DOCUMENTATION

The API documentation provides a detailed overview of the endpoints, request methods, required parameters, and responses for interacting with the Book Store App backend. All APIs adhere to RESTful principles and return responses in JSON format.

Authentication APIs

1. **Register User**
    - Endpoint: /auth/register
    - Method: POST
    - Description: Registers a new user in the system.

Request Body:

```
{
    "name": "John Doe",
    "email": "john.doe@example.com",
    "password": "securePassword123"
}
```

Response:

Success (201):

```
json
{
    "message": "User registered successfully",
    "user": {
        "id": "userId123",
        "name": "John Doe",
        "email": "john.doe@example.com"
    }
}
```

Error (400):

json

```
{
    "error": "Email already exists"
}
```

2. Login User

- Endpoint: /auth/login
- Method: POST
- Description: Authenticates a user and generates a JWT token.

Request Body:

json

```
{
    "email": "john.doe@example.com",
    "password": "securePassword123"
}
```

Response:

Success (200):

json

```
{
    "message": "Login successful",
    "token": "jwtToken123",
    "user": {
    "id": "userId123",
    "name": "John Doe",
    "email": "john.doe@example.com"
```

```
    }
}
```

Error (401):

```
json
{
    "error": "Invalid email or password"
}
```

3. Get All Books

- **Endpoint:** /books
- **Method:** GET
- **Description:** Retrieves a list of all books in the store.

Response:

Success (200):

```
json
[
  {
    "id": "bookId123",
    "title": "Learn MERN Stack",
    "author": "Jane Doe",
    "genre": "Programming",
    "price": 29.99,
    "stock": 100,
```

```
        "createdAt": "2024-11-23T09:00:00Z"
    },
    {
        "id": "bookId124",
        "title": "React for Beginners",
        "author": "John Smith",
        "genre": "Programming",
        "price": 19.99,
        "stock": 50,
        "createdAt": "2024-11-22T08:00:00Z"
    }
]
```


4. Get Book by ID

- **Endpoint:** /books/:id
- **Method:** GET
- **Description:** Retrieves details of a specific book by its ID.

Response:

Success (200):

json


```
{
    "id": "bookId123",
    "title": "Learn MERN Stack",
    "author": "Jane Doe",
```

```
"genre": "Programming",
"price": 29.99,
"stock": 100,
"reviews": [
    {
        "userId": "userId123",
        "rating": 5,
        "comment": "Excellent book!"
    }
],
"createdAt": "2024-11-23T09:00:00Z"
}
```


Error (404):


```
json
{
    "error": "Book not found"
}
```


5.Add New Book :

- **Endpoint:** /books
- **Method:** POST
- **Description:** Adds a new book to the inventory (admin access required).

Request Body:


```
json
{
```

```
"title": "Node.js Guide",
"author": "Sam Lee",
"genre": "Programming",
"price": 24.99,
"stock": 80
}
```

Response:

Success(201):

json

```
{
"message": "Book added successfully",
"book": {
"id": "bookId125",
"title": "Node.js Guide",
"author": "Sam Lee",
"genre": "Programming",
"price": 24.99,
"stock": 80
}
}
```

Error(403):

json

```
{
```

```
"error": "Unauthorized access"
}
```

6. Add Item to Cart

- **Endpoint:** /cart
- **Method:** POST
- **Description:** Adds an item to the user's cart.

Request Body:

```
json
{
"bookId": "bookId123",
"quantity": 2
}
```

Response:

Success(200):

```
json
{
"message": "Item added to cart",
"cart": {
"userId": "userId123",
"items": [
{
"bookId": "bookId123",
"quantity": 2
```

```
            }
        ]
    }
}
```

Error(400):

```
json
{
    "error": "Book is out of stock"
}
```

7.Get Cart Details

- **Endpoint:** /cart
- **Method:** GET
- **Description:** Retrieves the current user's cart details.

Response:

Success(200):

```
json
{
    "userId": "userId123",
    "items": [
        {
            "bookId": "bookId123",
            "quantity": 2,
            "title": "Learn MERN Stack",
            "price": 29.99
        }
    ],
    "totalPrice": 59.98
}
```

8. Place Order

- **Endpoint:** /orders
- **Method:** POST
- **Description:** Places an order for the items in the user's cart.

Request Body:

json

```
{
    "paymentMethod": "Credit Card",
     "shippingAddress": "123 Main St, Springfield"
}
```

Response:

Success (201):

json

```
{
     "message": "Order placed successfully",
      "order": {
              "id": "orderId123",
              "totalPrice": 59.98,
              "paymentStatus": "Paid",
              "deliveryStatus": "Processing"
       }
}
```

9. Get Sales Report (Admin Only)

- **Endpoint:** /admin/reports/sales
- **Method:** GET
- **Description:** Fetches a detailed sales report.

Response:

Success(200):

json

```
{
        "totalSales": 5000,
         "orders": 100,
         "booksSold": 150
 }
```

AUTHENTICATION & AUTHORIZATION

AUTHENTICATION

Authentication verifies the identity of users, ensuring that only valid users can access the system.

Implementation Steps:

1.User Registration:

- Endpoint: /auth/register
- Users provide details like name, email, and password.
- Password Hashing:

  Use bcrypt.js to hash passwords before storing them in the database.

2.**User Login:**

- Endpoint: /auth/login
- Users authenticate by providing their email and password.
- Validation Process:
  - Match the provided email with the database record.
  - Compare the submitted password with the stored hashed password using bcrypt.compare.

3.**Token Generation:**

- After successful login, generate a **JWT (JSON Web Token)**.
- JWT Structure:
  - Header: Specifies the algorithm and token type.
  - Payload: Contains user-specific data like userId and role.
  - Signature: Verifies the token's integrity using a secret key.

4.Token Storage:

- The JWT is sent to the client, typically stored in HTTP-only cookies or localStorage for secure and persistent access.

AUTHORIZATION

Authorization ensures that users can only access resources and perform actions they are permitted to.

Role-Based Access Control (RBAC):

1.Roles Definition:

Customer:

Browse books, manage the cart, and place orders.

Admin:

Manage inventory, view reports, and update order statuses.

2.Middleware for Role Validation:

- Create middleware to check user roles and restrict access to sensitive endpoints.

Security Best Practices

1.Password Security:

- Always hash passwords before storing.
- Use a strong hashing algorithm like bcrypt with sufficient rounds (e.g., saltRounds = 10).

2.Token Security:

- Use HTTP-only cookies for storing JWTs to prevent XSS attacks.
- Add token expiration (e.g., expiresIn: '1d') to limit token lifespan.
- Implement token blacklisting for logout functionality or compromised tokens.

3.Input Validation:

- Validate user inputs (e.g., email format, password strength) to prevent SQL injection or other attacks.

4.Access Control:

- Implement least privilege access by assigning minimal permissions based on roles.

5.Environment Variables:

- Store sensitive information like JWT secrets in environment variables.

6.Rate Limiting:

- Apply rate limiting (e.g., using express-rate-limit) to prevent brute force attacks on login.

TESTING

Unit Testing with Jest (for Book App Store - Cart Functionality)

Cart Functionality Code (cart.js):

```
  let cart = [];
  const addToCart = (product) => {
   cart.push(product);
};

const getTotalPrice = () => {
   return cart.reduce((total, item) => total + item.price, 0);
};

const getCart = () => cart;

const clearCart = () => {
   cart = [];
};

module.exports = { addToCart, getTotalPrice, getCart, clearCart };
```

Unit Test Code for cart.js (cart.test.js):

```
const { addToCart, getTotalPrice, getCart, clearCart } = require('./cart');


describe('Cart Functionality', () => {
    beforeEach(() => {
        clearCart(); // Clear the cart before each test
    });

    test('should add a product to the cart', () => {
        const product = { id: 1, name: 'Book A', price: 10 };
        addToCart(product);

        const cart = getCart();
        expect(cart).toHaveLength(1);
        expect(cart[0]).toEqual(product);
    });

    test('should calculate the total price of products in the cart', () => {
        addToCart({ id: 1, name: 'Book A', price: 10 });
        addToCart({ id: 2, name: 'Book B', price: 20 });

        const totalPrice = getTotalPrice();
        expect(totalPrice).toBe(30);
```

```
  });
  test('should return an empty cart after clearing it', () => {
    addToCart({ id: 1, name: 'Book A', price: 10 });
    clearCart();

    const cart = getCart();
    expect(cart).toHaveLength(0);
  });
});
```

Running Unit Tests

1.Install Jest:

```
npm install --save-dev jest
```

2. Add a test script in package.json:

```
"scripts": {
    "test": "jest"
}
```

3.Run the tests:

```
npm test
```

Security Testing: SQL Injection Prevention

Simulating SQL Injection (security.test.js)

```
const { queryDatabase } = require('./db'); // Simulate database query function


test('should not allow SQL injection in queries', async () => {
  const maliciousInput = "1 OR 1=1"; // SQL Injection attempt
  const query = `SELECT * FROM products WHERE id = '${maliciousInput}'`;


  const result = await queryDatabase(query);
  expect(result).toBeNull(); // Ensure no results or proper handling of malicious input
});
```

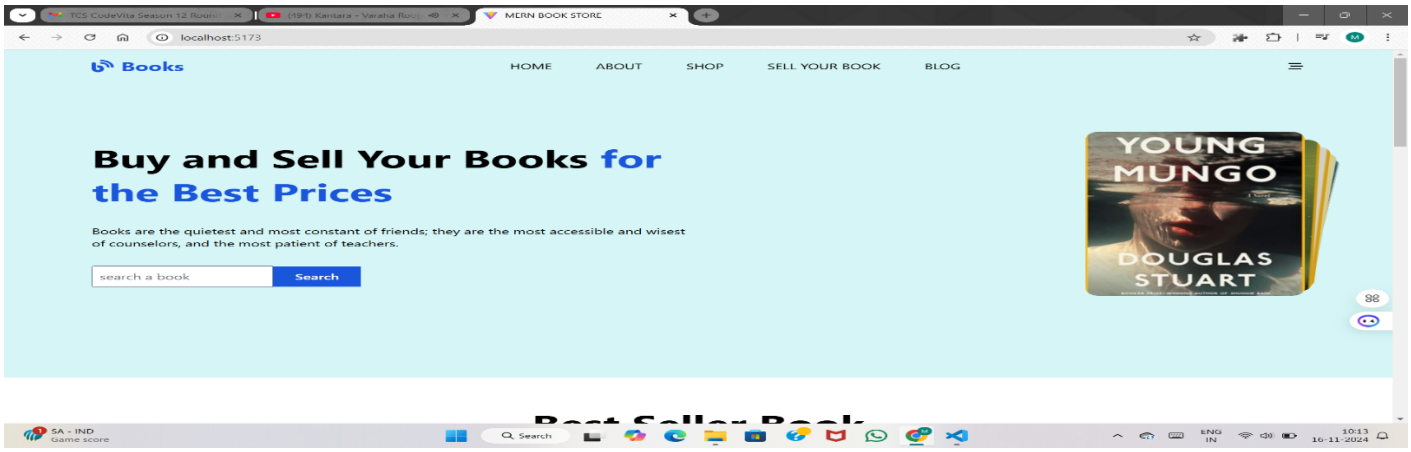
SQL Injection Protection Implementation

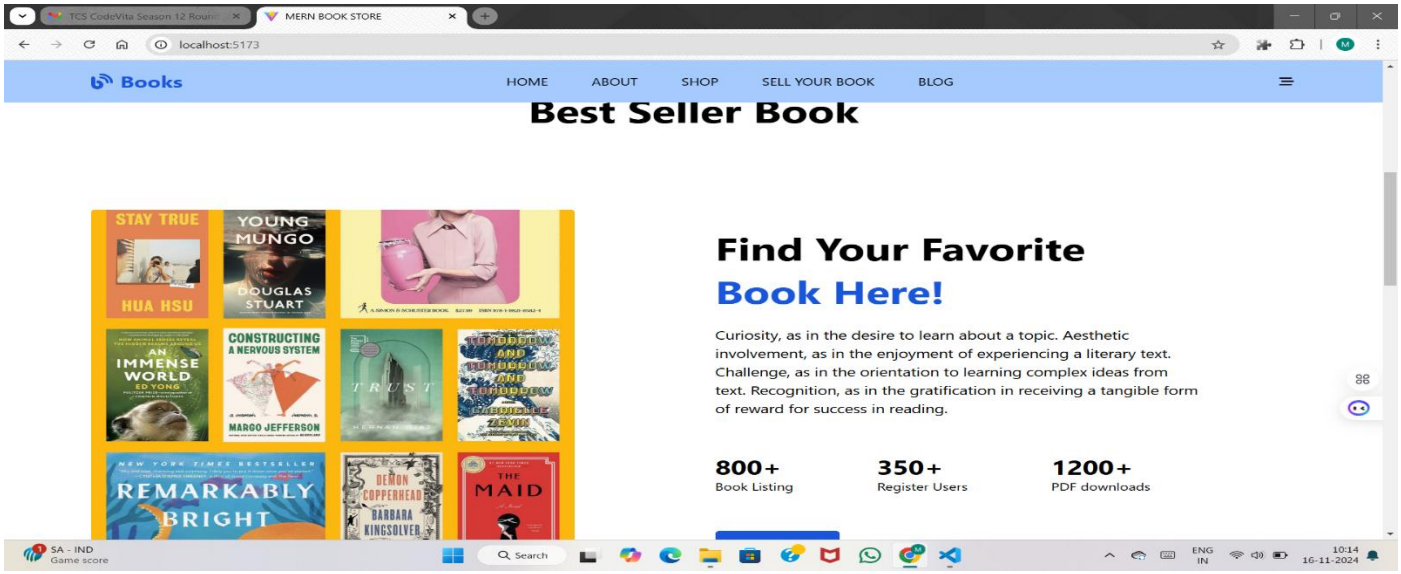
- **Sanitize Inputs:** Use parameterized queries or ORM libraries like Sequelize or Mongoose.

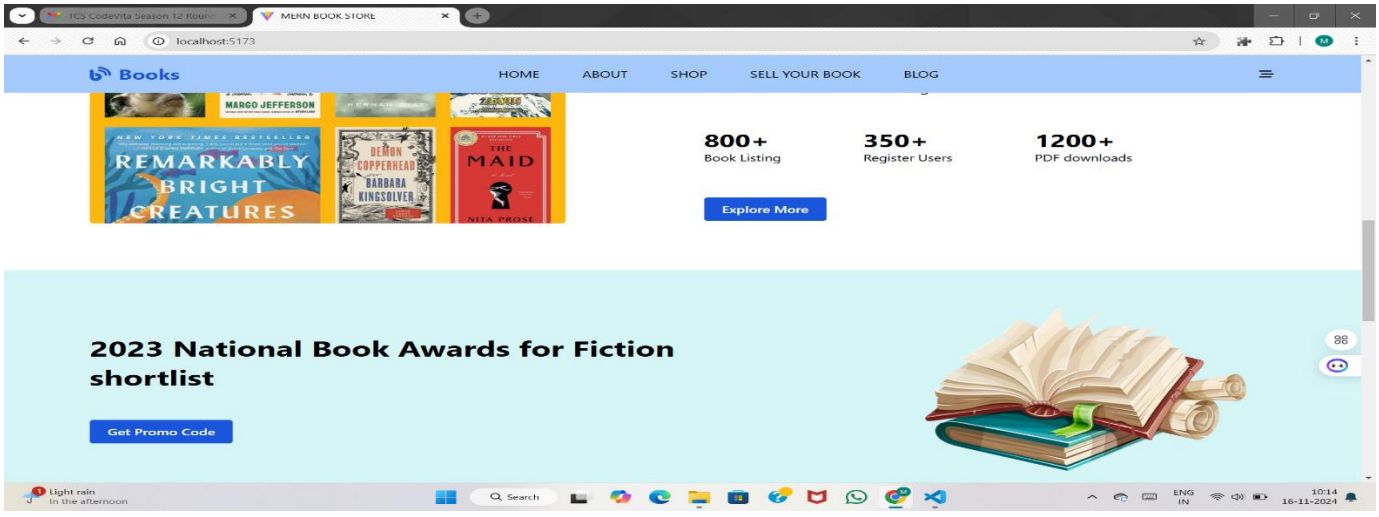
  Example (using Sequelize):

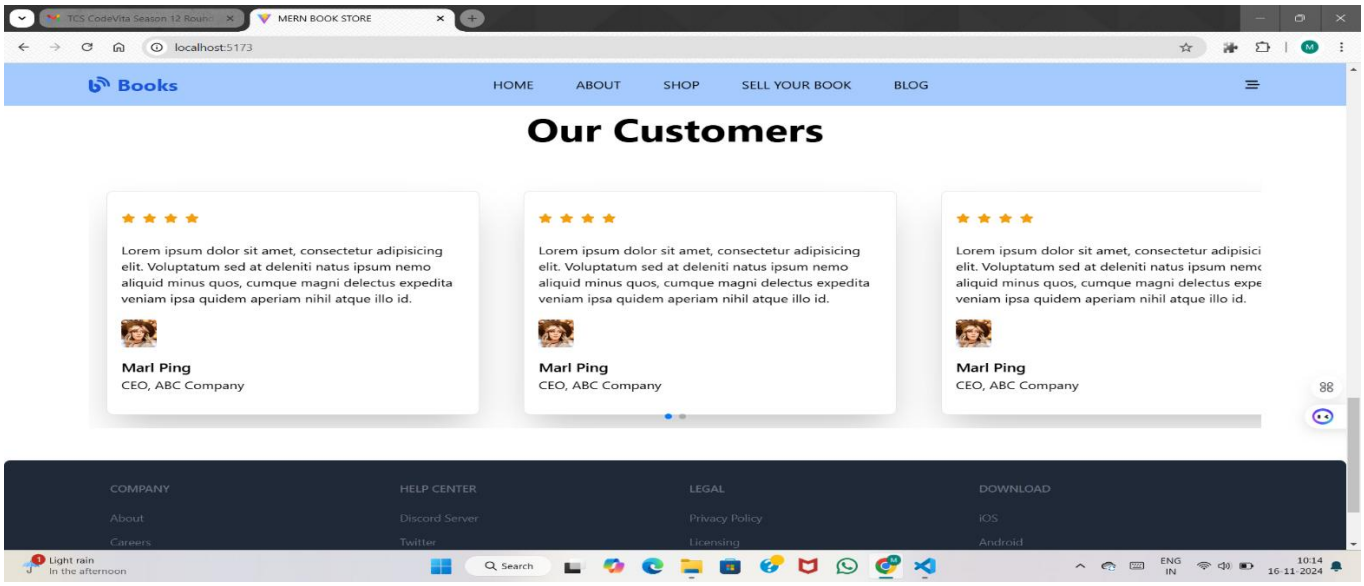
  ```
  const result = await Product.findOne({ where: { id: maliciousInput } });
  ```

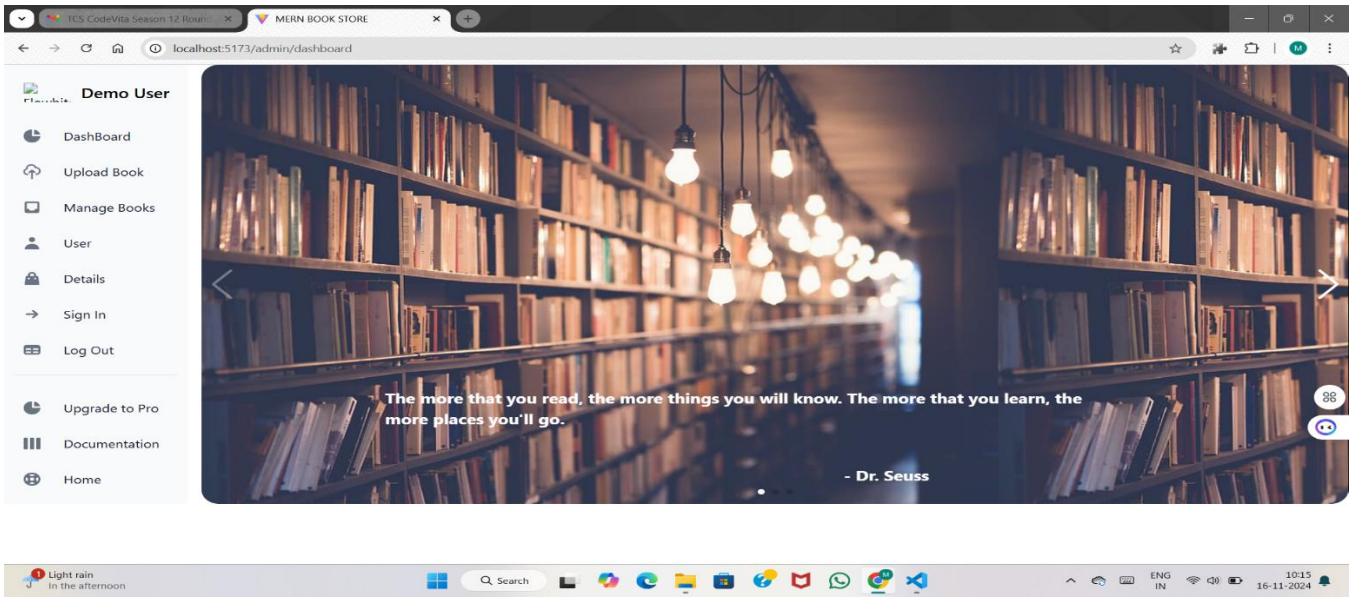
SCREENSHOTS

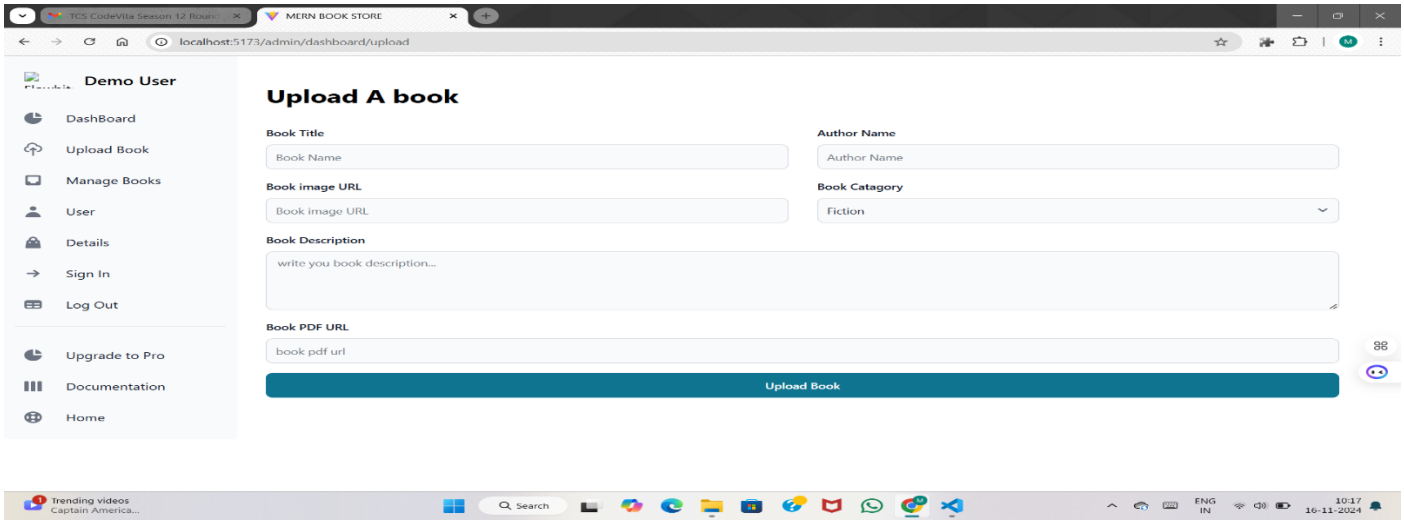


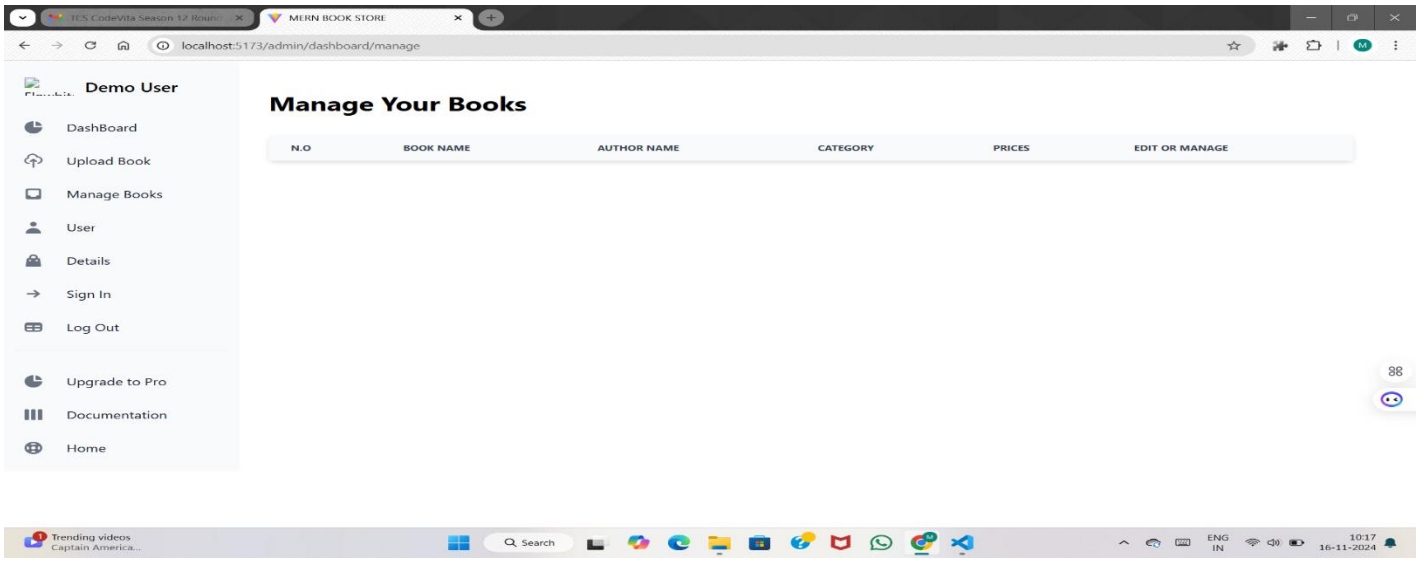


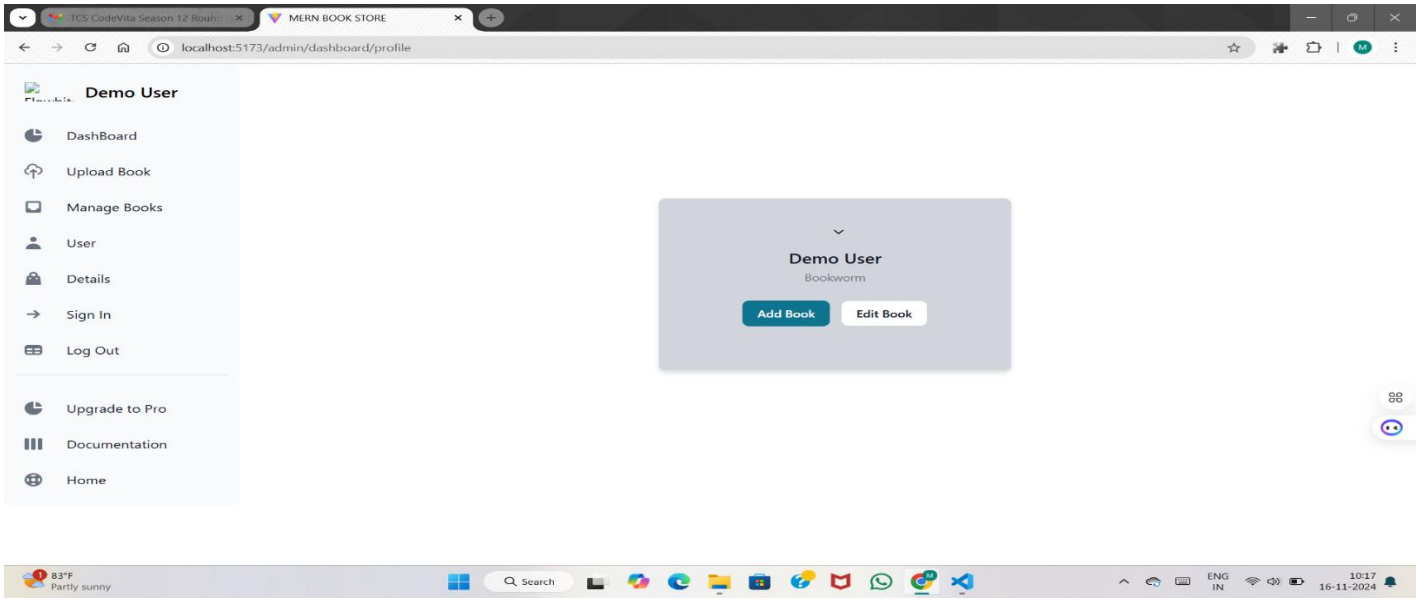


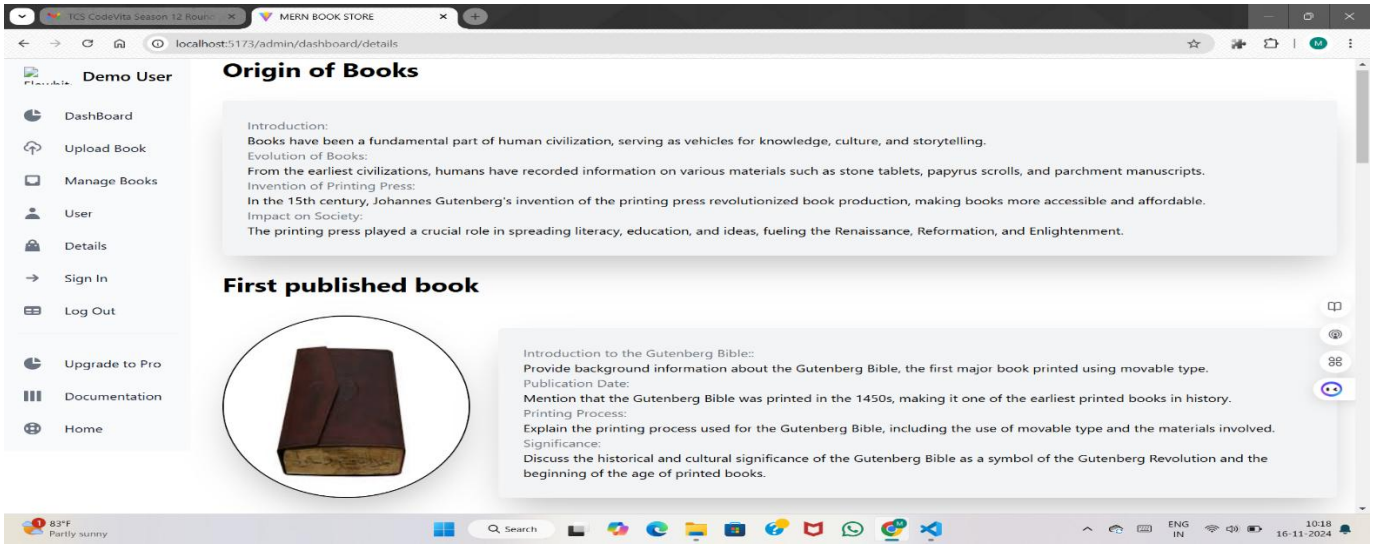

MERN BOOK STORE
localhost:5173
Books
HOME
ABOUT
SHOP
SELL YOUR BOOK
BLOG
Our Customers
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptatum sed at deleniti natus ipsum nemo aliquid minus quos, cumque magni delectus expedita veniam ipsa quidem aperiam nihil atque illo id.
Marl Ping
CEO, ABC Company
COMPANY
About
Careers
HELP CENTER
Discord Server
Twitter
LEGAL
Privacy Policy
Licensing
DOWNLOAD
iOS
Android

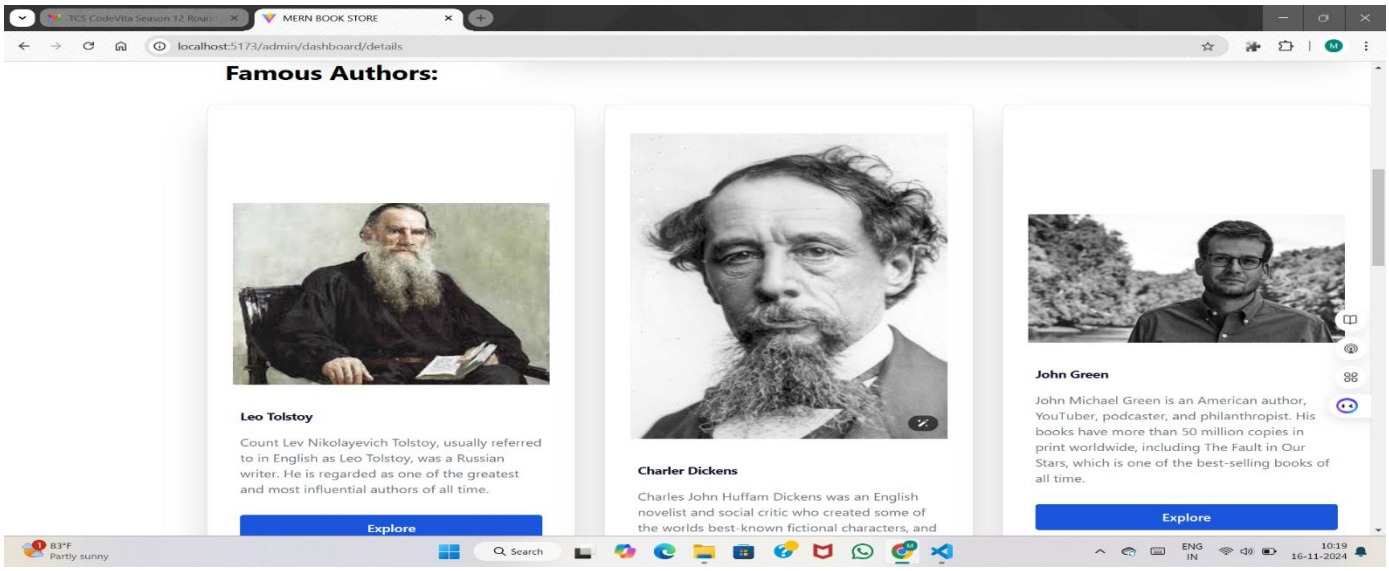

MERN BOOK STORE
localhost:5173/admin/dashboard
Demo User
DashBoard
Upload Book
Manage Books
User
Details
Sign In
Log Out
Upgrade to Pro
Documentation
Home
The more that you read, the more things you will know. The more that you learn, the more places you'll go.
- Dr. Seuss

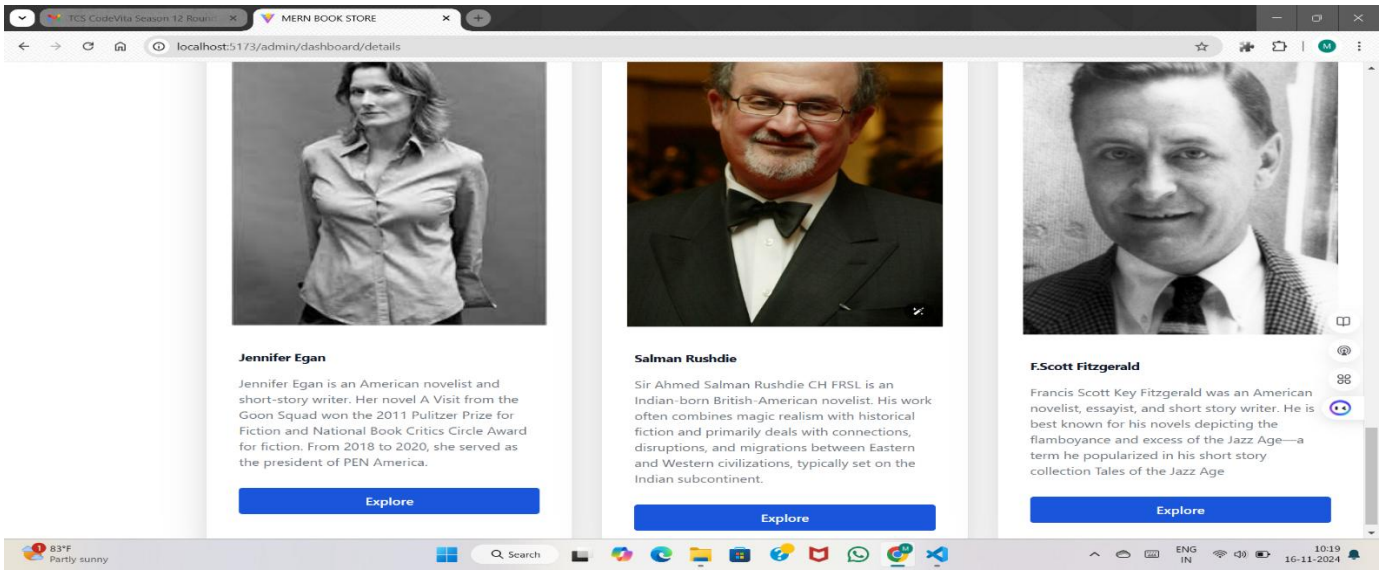

MERN BOOK STORE
localhost:5173/admin/dashboard/upload
Demo User
DashBoard
Upload Book
Manage Books
User
Details
Sign In
Log Out
Upgrade to Pro
Documentation
Home
Upload A book
Book Title
Book Name
Author Name
Author Name
Book image URL
Book image URL
Book Catagory
Fiction
Book Description
write you book description...
Book PDF URL
book pdf url
Upload Book

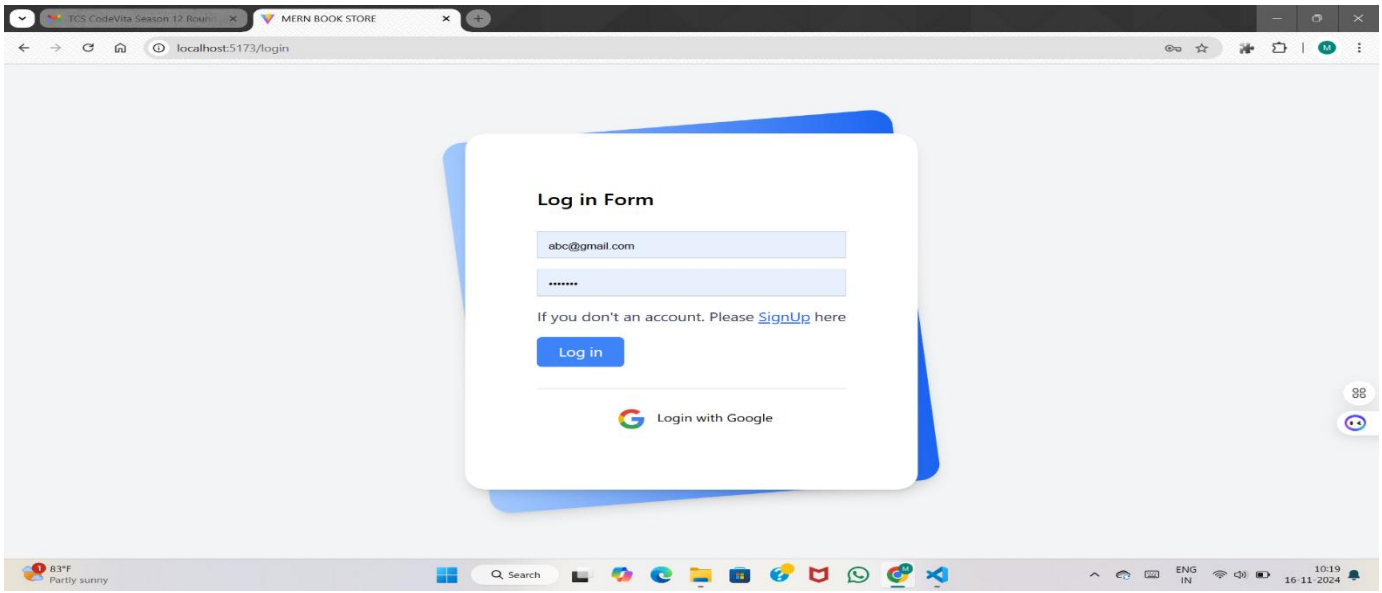

MERN BOOK STORE
localhost:5173/admin/dashboard/manage
Demo User
DashBoard
Upload Book
Manage Books
User
Details
Sign In
Log Out
Upgrade to Pro
Documentation
Home
Manage Your Books
N.O
BOOK NAME
AUTHOR NAME
CATEGORY
PRICES
EDIT OR MANAGE


MERN BOOK STORE
localhost:5173/admin/dashboard/profile
Demo User
DashBoard
Upload Book
Manage Books
User
Details
Sign In
Log Out
Upgrade to Pro
Documentation
Home
Demo User
Bookworm
Add Book
Edit Book


MERN BOOK STORE
localhost:5173/admin/dashboard/details
Demo User
DashBoard
Upload Book
Manage Books
User
Details
Sign In
Log Out
Upgrade to Pro
Documentation
Home
Origin of Books
Introduction:
Books have been a fundamental part of human civilization, serving as vehicles for knowledge, culture, and storytelling.
Evolution of Books:
From the earliest civilizations, humans have recorded information on various materials such as stone tablets, papyrus scrolls, and parchment manuscripts.
Invention of Printing Press:
In the 15th century, Johannes Gutenberg's invention of the printing press revolutionized book production, making books more accessible and affordable.
Impact on Society:
The printing press played a crucial role in spreading literacy, education, and ideas, fueling the Renaissance, Reformation, and Enlightenment.
First published book
Introduction to the Gutenberg Bible::
Provide background information about the Gutenberg Bible, the first major book printed using movable type.
Publication Date:
Mention that the Gutenberg Bible was printed in the 1450s, making it one of the earliest printed books in history.
Printing Process:
Explain the printing process used for the Gutenberg Bible, including the use of movable type and the materials involved.
Significance:
Discuss the historical and cultural significance of the Gutenberg Bible as a symbol of the Gutenberg Revolution and the beginning of the age of printed books.


MERN BOOK STORE
localhost:5173/admin/dashboard/details
Famous Authors:
Leo Tolstoy
Count Lev Nikolayevich Tolstoy, usually referred to in English as Leo Tolstoy, was a Russian writer. He is regarded as one of the greatest and most influential authors of all time.
Explore
Charler Dickens
Charles John Huffam Dickens was an English novelist and social critic who created some of the worlds best-known fictional characters, and
John Green
John Michael Green is an American author, YouTuber, podcaster, and philanthropist. His books have more than 50 million copies in print worldwide, including The Fault in Our Stars, which is one of the best-selling books of all time.
Explore


MERN BOOK STORE
localhost:5173/admin/dashboard/details
Jennifer Egan
Jennifer Egan is an American novelist and short-story writer. Her novel A Visit from the Goon Squad won the 2011 Pulitzer Prize for Fiction and National Book Critics Circle Award for fiction. From 2018 to 2020, she served as the president of PEN America.
Explore
Salman Rushdie
Sir Ahmed Salman Rushdie CH FRSL is an Indian-born British-American novelist. His work often combines magic realism with historical fiction and primarily deals with connections, disruptions, and migrations between Eastern and Western civilizations, typically set on the Indian subcontinent.
Explore
F.Scott Fitzgerald
Francis Scott Key Fitzgerald was an American novelist, essayist, and short story writer. He is best known for his novels depicting the flamboyance and excess of the Jazz Age—a term he popularized in his short story collection Tales of the Jazz Age
Explore


MERN BOOK STORE
localhost:5173/login
Log in Form
abc@gmail.com
If you don't an account. Please SignUp here
Log in
Login with Google

KNOWN ISSUES

While the Book App Store is functional and provides essential features for users, there may still be areas requiring improvement or optimization. Below is a list of potential known issues categorized by type.

1. User Experience (UX) Issues

- **Search Functionality:**
  The search bar lacks advanced filtering options such as genre, author, price range, or publication year.
- **Pagination in Book Listings:**
  Large datasets for book listings may slow down the UI due to the lack of server-side pagination.
- **Error Messages:**
  Error messages are generic (e.g., "Something went wrong"), making it hard for users to understand specific issues like failed payments or incorrect inputs.
- **Responsiveness on Older Devices:**
  While the app is responsive, performance and rendering issues have been reported on older mobile devices or browsers.

2. Backend/API Issues

- **Rate Limiting:**
  The APIs do not enforce rate-limiting, making them vulnerable to abuse or potential denial-of-service (DoS) attacks.
- **Caching:**
  Lack of caching for frequently accessed data, such as book details, results in slower response times for repeated requests.

- **Error Handling:**
  Some API endpoints do not return consistent error structures, which may confuse the frontend during error state handling.
- **Bulk Uploads:**
  No support for bulk upload of book data (e.g., CSV or Excel), which complicates managing inventory for admins.

3. Database Issues

- **Scaling for High Traffic:**
  The current database design may struggle with high traffic or large datasets, particularly for querying large book inventories.
- **Redundant Data:**
  Some database queries involve redundant data storage, leading to unnecessary increases in storage usage.
- **Search Performance:**
  Search queries are not optimized for large-scale data, leading to slower performance as the database grows.

4. Security Issues

- **Token Expiry Management:**
  Users remain logged in indefinitely, as token expiration handling and auto-refresh mechanisms are incomplete.
- **CSRF Protection:**
  Some critical forms (e.g., payment submission) do not have robust Cross-Site Request Forgery (CSRF) protection mechanisms.

- **Password Security:**
  Although passwords are hashed, no advanced measures like password salting or encryption during transmission are implemented.
- **Error Disclosure:**
  Error messages sometimes expose stack traces or sensitive details that attackers could exploit.

5. Payment Issues

- **Payment Gateway Errors:**
  Payments occasionally fail due to unreliable integration with the third-party payment gateway, but users are not notified of the root cause.
- **Currency Handling:**
  Multi-currency support is limited, with prices defaulting to one currency, leading to user confusion in international transactions.

FUTURE ENHANCEMENTS

To ensure the Book App Store stays competitive and meets evolving user expectations, several potential enhancements can be implemented. These enhancements aim to improve functionality, user experience, scalability, and profitability.

1. Advanced Search and Filtering

- **Multi-Parameter Search:**
  - Enable search by title, author, genre, language, publication year, and price range.
- **Search Suggestions:**
  - Implement an autocomplete feature to provide suggestions as users type.
- **Voice Search:**
  - Integrate voice-based search for hands-free convenience.
- **AI-Powered Recommendations:**
  - Use AI to suggest books based on user preferences, browsing history, or ratings.

2. Personalization Features

- **Customized Homepages:**
  - Show personalized recommendations on the homepage based on user interests.
- **Wishlist and Favorites:**
  - Allow users to create and share wishlists for books they want to buy later.
- **Reading History:**

- Display a history of previously purchased or viewed books for user reference.

3. Enhanced User Engagement

- **Book Clubs and Discussions:**
  - Add forums or virtual book clubs where users can discuss books and share reviews.
- **Gamification:**
  - Introduce badges, rewards, or loyalty points for frequent purchases, reviews, or referrals.
- **Book Previews:**
  - Provide free previews or sample chapters for users to explore before purchasing.

4. Multi-Currency and Localization Support

- **Multi-Currency Payments:**
  - Allow users to view prices and pay in their local currencies.
- **Localization:**
  - Translate the app into multiple languages for a global audience.
- **Localized Content:**
  - Curate book recommendations based on regional preferences.

5. Integration with External Platforms

- **Social Media Integration:**
    - Enable users to share their favorite books or reviews directly to social platforms.
- **Integration with E-Readers:**
    - Allow purchased eBooks to be seamlessly integrated with Kindle, Google Books, or other e-reader platforms.
- **Affiliate Marketing:**
    - Partner with publishers or authors to promote exclusive content or offers.

6. Mobile App Development

- **Native Mobile App:**
    - Create dedicated Android and iOS apps for better performance and offline capabilities.
- **Push Notifications:**
    - Notify users about discounts, new arrivals, or wishlisted book availability.

7. Enhanced Payment Options

- **Flexible Payment Methods:**
    - Introduce support for Buy Now, Pay Later (BNPL) services like Afterpay or Klarna.
- **Cryptocurrency Payments:**
    - Allow payments via popular cryptocurrencies like Bitcoin or Ethereum.