

NAAN MDUHALVAN PROJECT REPORT

TITLE :BOOK STORE - MERN

COLLEGE: AALIM MUHAMMED SALEGH COLLEGE OF ENGINEERING

MEMBERS:

- 1.NAJEEBULLAH N (110121104070)
- 2.MOHAMMED SOWBAN (110121104311)
- 3.MOHAMMED IRZATH (110121104052)
- 4.MOHAMMED UWAIS KHADER SHAIK (110121104067)

BOOK STORE - MERN

INTRODUCTION:

The Book Store application is a comprehensive online platform developed to facilitate easy access to a variety of books. It aims to connect book enthusiasts with a vast selection of genres, including fiction, non-fiction, academic texts, and more. Users can explore, purchase, and review books with a streamlined, user-friendly interface. This application serves as a one-stop solution for book lovers, offering features like book search, cart management, user profiles, and order tracking.

DESCRIPTION:

Welcome to the Book Store, a modern platform tailored for book lovers. Our Book Store offers a curated selection of books from multiple genres and authors, giving users the freedom to browse, read summaries, and make informed purchase decisions. The Book Store application features:

- ❑ **User-friendly Search:** Advanced search capabilities that allow users to filter books by title, author, genre, and price.
- ❑ **Seamless Purchase Process:** Users can add books to a cart, proceed with a secure checkout, and receive order updates.
- ❑ **Personalized Profiles:** Each user has a profile to manage their order history, wishlist, and personalized book recommendations.
- ❑ **Ratings and Reviews:** Users can share feedback and ratings for books, helping others make better choices.

This Book Store project showcases the capabilities of the **MERN** stack (MongoDB, Express, React, Node.js) in creating a dynamic, efficient, and interactive web application.

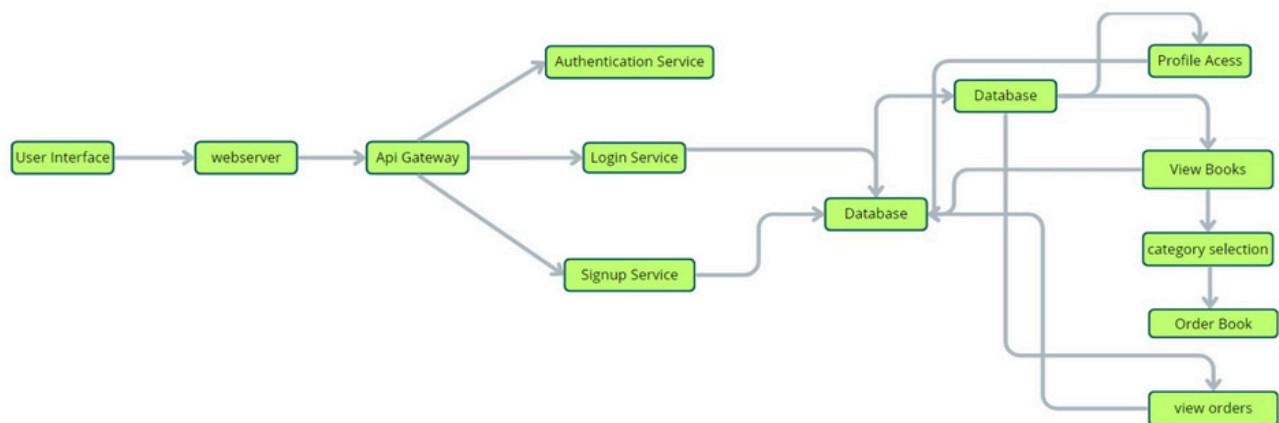
Scenario-based case-study:

Case Study Example:

John is an avid reader who loves to collect books. While looking for an easy way to discover new books and buy them online, he discovers our Book Store platform. Here's how John uses it:

1. **Finding the Perfect Book:** John logs into the platform and explores books using the genre filter, browsing fiction novels. He finds a recommended book titled "*The Alchemist*" and reads the synopsis.
2. **Adding to Cart:** Convinced by the reviews, John adds the book to his cart, continues browsing, and finds two more books to buy.
3. **Seamless Checkout:** John proceeds to checkout, where he selects his preferred payment method, completes the transaction, and receives an order confirmation email.
4. **Tracking and Reviewing:** After reading the book, John returns to the platform to leave a positive review. His experience encourages him to return for more books.

TECHNICAL ARCHITECTURE:

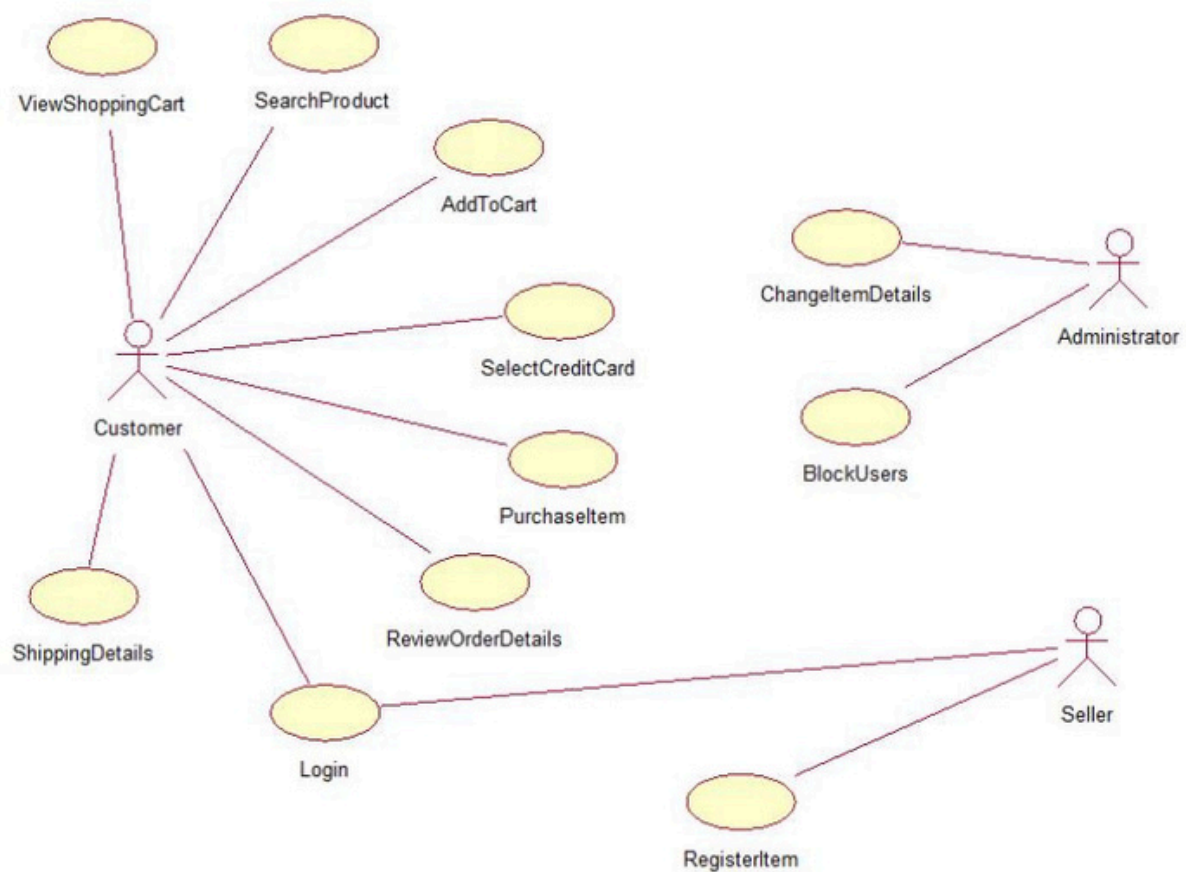


The Book Store application follows a client-server architecture:

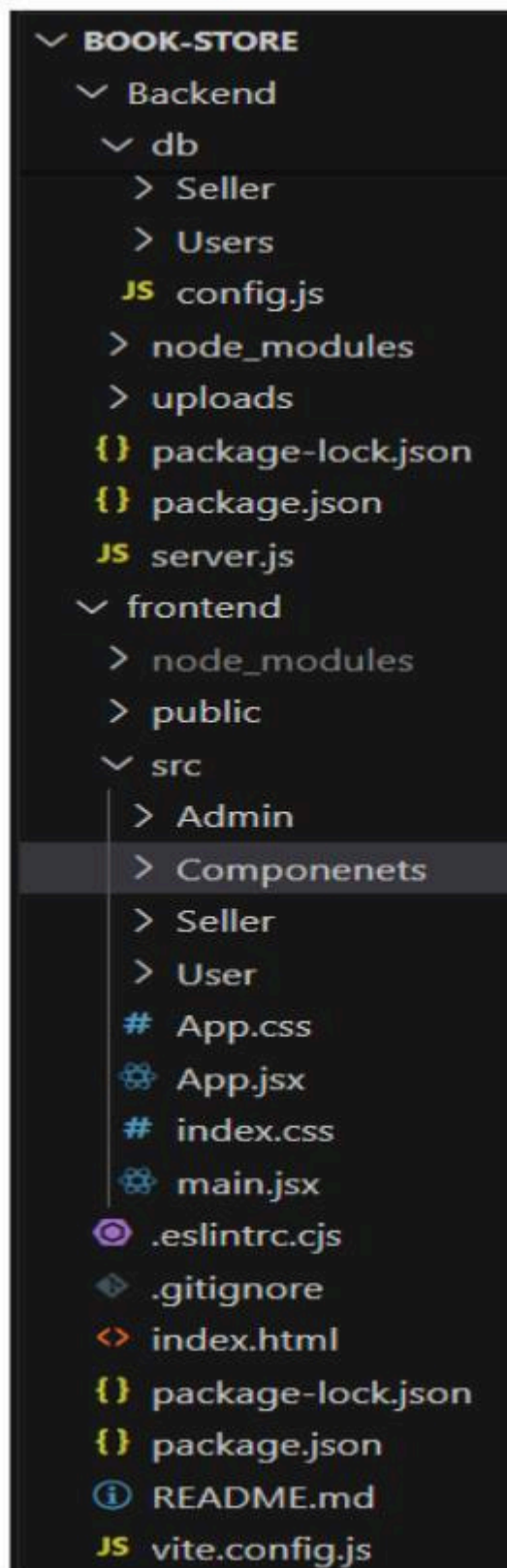
Frontend (Client): Developed using React.js, the frontend offers an interactive UI for users. React components manage the interface for browsing books, user accounts, and checkout processes. For asynchronous operations, Axios communicates with the backend via RESTful APIs.

- **Backend (Server):** Built on Node.js and Express, the backend handles all business logic, including book data management, user authentication, and order processing.
- **Database (MongoDB):** MongoDB stores structured data, such as book information, user details, orders, and reviews.

ER DIAGRAM:



PROJECT STRUCTURE:



- ❑ **Client:** Contains the React application, structured into components like book listings, cart, and user profile. Redux or Context API may be used to manage application state.
- ❑ **Server:** Houses the Express application with defined routes for handling requests, middleware for authentication, and Mongoose models for database interactions.
- ❑ **Database:** MongoDB collections include Users, Books, Orders, and Reviews.

PRE-REQUISITIC:

Here 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:

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/>

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/>

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**

React js: React is a JavaScript library for building client-side applications.

And Creating Single Page Web-Appliacion

Getting Started

Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

Quik Start

```
npm create vite@latest
```

```
cd my-app
```

```
npm install
```

```
npm run dev
```

If you've previously installed create-react-app globally via `npm install -g create-react-app`, we recommend you uninstall the package using `npm uninstall -g create-react-app` or `yarn global remove create-react-app` to ensure that npx always uses the latest version.

Create a new React project:

- Choose or create a directory where you want to set up your React project.
- Open your terminal or command prompt.
- Navigate to the selected directory using the `cd` command.
- Create a new React project by running the following command: `npx create-react-app your-app-name`. Wait for the project to be created:
- This command will generate the basic project structure and install the necessary dependencies

Navigate into the project directory:

- After the project creation is complete, navigate into the project directory by running the following command: **`cd your-app-name`**

Start the development server:

- To launch the development server and see your React app in the browser, run the following command: **npm run dev**
- The npm start will compile your app and start the development server.
- Open your web browser and navigate to <https://localhost:5173> to see your React app.

You have successfully set up React on your machine and created a new React project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for React. You can explore more

ad-

vanced configurations and features by referring to the official React documentation:

<https://react.dev/> **HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the

structure of your

app, CSS for styling, and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM)

library like Mongoose to connect your Node.js server with the MongoDB database and perform

CRUD (Create, Read, Update, Delete) operations. **Front-end Library:** Utilize React to build

the user-facing part of the application, including

products listings, booking forms, and user interfaces for the admin dashboard.

Version Control: Use Git for version control, enabling collaboration and tracking changes

throughout the development process. Platforms like GitHub or Bitbucket can host your

repository. • Git: Download and installation instructions can be found at: [https://git-](https://git-scm.com/downloads)

[scm.com/downloads](https://git-scm.com/downloads) **Development Environment:** Choose a code editor or Integrated

Development Environment

(IDE) that suits your preferences, such as [Visual Studio Code](#), [Sublime Text](#), or [WebStorm](#).

- Visual Studio Code: Download from <https://code.visualstudio.com/download>

- Sublime Text: Download from <https://www.sublimetext.com/download>

- WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

Application flow:

Customer Responsibilities:

- **Book Selection:** Customers can browse a diverse collection of books, filter by categories, and read detailed descriptions before selecting.
- **Order Placement:** Once a book is selected, customers can add it to their cart and proceed to check out. They need to review their order and ensure accuracy before confirming.
- **Timely Payment:** Customers are responsible for making secure payments for the books they purchase through the payment gateway provided.
- **Effective Communication:** If any assistance is needed, customers should communicate with the support team for timely resolutions or clarifications.
- **Feedback and Reviews:** After reading a purchased book, customers are encouraged to provide honest reviews and ratings, contributing to a supportive community for other readers.

Store Manager Responsibilities:

- ☐ **Inventory Management:** Store managers are responsible for keeping an updated catalog, ensuring book listings are accurate, and promptly updating any changes, such as new arrivals or out-of-stock items.
- ☐ **Order Fulfillment:** After a customer place an order, the store manager oversees order fulfillment, packing, and delivery arrangements to ensure timely delivery.
- ☐ **Promotional Activities:** Managers may initiate discounts or special offers to enhance customer engagement and sales. Promotions should be clear and accurately represented in the application.
- ☐ **Customer Communication:** The store manager should promptly respond to customer queries and resolve issues, providing a positive customer experience.
- ☐ **Quality Control:** All book descriptions, cover images, and related content should be accurate and reviewed to maintain high quality in the catalog and avoid customer dissatisfaction.

Admin Responsibilities:

- ☐ **Data Oversight:** The admin oversees all data stored in the application, including customer data, book inventory, order details, and transactions, ensuring data integrity and security.
- ☐ **Policy Enforcement:** Admins enforce application policies regarding usage, data privacy, and transactional integrity to maintain a safe and ethical platform.
- ☐ **User Support and Communication:** Admins should facilitate seamless communication between customers and store managers, addressing escalated issues or complaints.
- ☐ **Platform Maintenance:** Admins are responsible for keeping the application functional, updating software or backend frameworks, and managing regular maintenance tasks.
- ☐ **Analytics and Reporting:** Admins should track key metrics such as sales trends, customer engagement, and inventory flow, using insights to improve platform functionality and user satisfaction.

Project Flow:

Milestone 1: Project setup and configuration.

✓ Folder setup:

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

- Client folders.
- Server folders

✓ Installation of required tools:

1. Open the frontend folder to install the necessary tools
for frontend, we use:

- Axios.
- React-Router –dom.
- Bootstrap.
- React-Bootstrap.

2. Open the backend folder to install the necessary tools

for backend, we use:

- Express.
- Mongoose.
- Cors.

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.

```
package.json M X
client > package.json > ...
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.5.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.11.0",
    "react-router-dom": "^6.19.0",
    "react-scripts": "5.0.1",
    "socket.io-client": "^4.7.2",
    "uuid": "^9.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.

```
package.json X
server > package.json > ...
  {
    "name": "server",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    "type": "module",
    "scripts": {
      "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "dependencies": {
      "bcrypt": "^5.1.1",
      "body-parser": "^1.20.2",
      "cors": "^2.8.5",
      "express": "^4.18.2",
      "http": "^0.0.1-security",
      "mongoose": "^7.6.1",
      "socket.io": "^4.7.2",
      "uuid": "^9.0.1"
    }
  }
```

Milestone 2: Backend Development

1. Setup express server

- Create index.js file in the server (backend folder).
- Create a .env file and define port number to access it globally.
- Configure the server by adding cors, body-parser.

2. User Authentication:

- Set up a MongoDB database (locally or using a cloud service like MongoDB Atlas).
- Create collections for:
 - Users (storing user information, account type)
 - Books (Book details, budget)
- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

3. API Routes:

- Create separate route files for different API functionalities such as user's orders, and authentication.
- Define the necessary routes for listing products, handling user registration and login, managing orders, etc.
- Implement route handlers using Express.js to handle requests and interact with the database.

4. Implement Data Models

- Define Mongoose schemas for the different data entities like products, users, and orders.
- Create corresponding Mongoose models to interact with the MongoDB database.
- Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

5. Error Handling:

- Implement errors handling middleware to catch and handle any errors that occur during the API requests.
- Return appropriate error responses with relevant error messages and HTTP status codes.

6. Admin Panel (Optional):

- Implement an admin panel with functionalities like:
 - Managing users
 - Monitoring Books updates and making a payment gateway securely

- Accessing transaction history

Milestone 3: Database development

1. Configure MongoDB:

- Install Mongoose.
- Create database connection.
- Create Schemas & Models.

2. Connect database to backend:

Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks like the one provided below.

```
const server = http.createServer(app);

const io = new Server(server, {
  cors: {
    origin: '*',
    methods: ['GET', 'POST', 'PUT', 'DELETE']
  }
});

io.on("connection", (socket) =>{
  console.log("User connected");

  SocketHandler(socket);
})

const PORT = 6001;

mongoose.connect('mongodb://localhost:27017/Freelancing',{
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(()=>{
```

```
  server.listen(PORT, ()=>{
    console.log(`Running @ ${PORT}`);
  });
}).catch((e)=> console.log(`Error in db connection ${e}`));
```

The Schemas for the database are given below:

The image displays two screenshots of a Visual Studio Code editor showing the MongoDB schemas for a project named 'Book-store-mernstack'.

Top Screenshot: user.model.js

```
backend > src > users > user.model.js > userSchema > username
1 const mongoose = require('mongoose')
2 const bcrypt = require('bcrypt');
3
4 const userSchema = new mongoose.Schema({
5   username: {
6     type: String,
7     required: true,
8     unique: true
9   },
10  password: {
11    type: String,
12    required: true
13  },
14  role: {
15    type: String,
16    enum: ['user', 'admin'],
17    required: true
18  }
19 })
20
21 userSchema.pre('save', async function( next ) {
22   if(!this.isModified('password')) return next();
23   this.password = await bcrypt.hash(this.password, 10);
24   next();
25 })
```

Bottom Screenshot: order.model.js

```
backend > src > orders > order.model.js > orderSchema
1 const mongoose = require('mongoose');
2
3 const orderSchema = new mongoose.Schema({
4   name: {
5     type: String,
6     required: true,
7   },
8   email: {
9     type: String,
10    required: true,
11  },
12  address: {
13    city: {
14      type: String,
15      required: true,
16    },
17    country: String,
18    state: String,
19    zipcode: String,
20  },
21  phone: {
22    type: Number,
23    required: true,
24  },
25 })
```

Milestone 4: Frontend development

1. Setup React Application:

- Create React application.
- Configure Routing.
- Install required libraries.

2. Design UI components:

- Create Components.
- Implement layout and styling.
- Add navigation.

3. Implement frontend logic:

- Integration with API endpoints.
- Implement data binding.

Milestone 5: Project Implementation

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the images provided below.

Landing page:



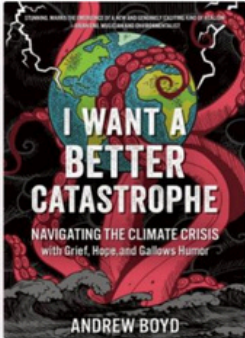
Search here

0

New Releases This Weeks

It's time to update your reading list with some of the latest and greatest releases in the literary world. From heart-pumping thrillers to captivating memoirs, this week's new releases offer something for everyone


Subscribe




Top Sellers

Choose a genre

Top 10 Fiction Books This Year



Mastering SEO in 2024



Best eCommerce Platforms

Authentication:

Search here

0

Please Login

Email


Email Address

Password

Password

Login

Haven't an account? Please [Register](#)

 Sign in with Google

©2025 Book Store. All rights reserved.

Register Page:

Search here

0

Please Register

Email

Email Address

Password

Password

Register

Have an account? Please [Login](#)

G

 Sign in with Google

©2025 Book Store. All rights reserved.

Admin Dashboard:

Cluster0 Data | Cloud: MongoD

Ecommerce BookStore

build-full-stack-book-store-me

localhost:5173/admin

localhost:5173 says

Admin Login successful!

OK

Admin Dashboard Login

Username

admin

Password

Login

©2025 Book Store. All rights reserved.

Add New Book:

The screenshot shows a web browser window with the URL `localhost:5173/dashboard/add-new-book`. The page features a dark sidebar on the left with icons for home, dashboard, and settings. The main content area has a header with 'Dashboard' and 'Book Store Inventory'. On the right, there are two buttons: 'Manage Books' and '+ Add New Book'. The central form is titled 'Add New Book' and contains the following fields:

- Title:** A text input field with the placeholder 'Enter book title'.
- Description:** A text input field with the placeholder 'Enter book description'.
- Category:** A dropdown menu with the selected option 'Choose A Category'.
- Trending:** An unchecked checkbox.
- Old Price:** A text input field with the placeholder 'Old Price'.
- New Price:** A text input field with the placeholder 'New Price'.
- Cover Image:** A file selection area with a 'Choose File' button and the text 'No file chosen'.

At the bottom of the form is a green button labeled 'Add Book'.

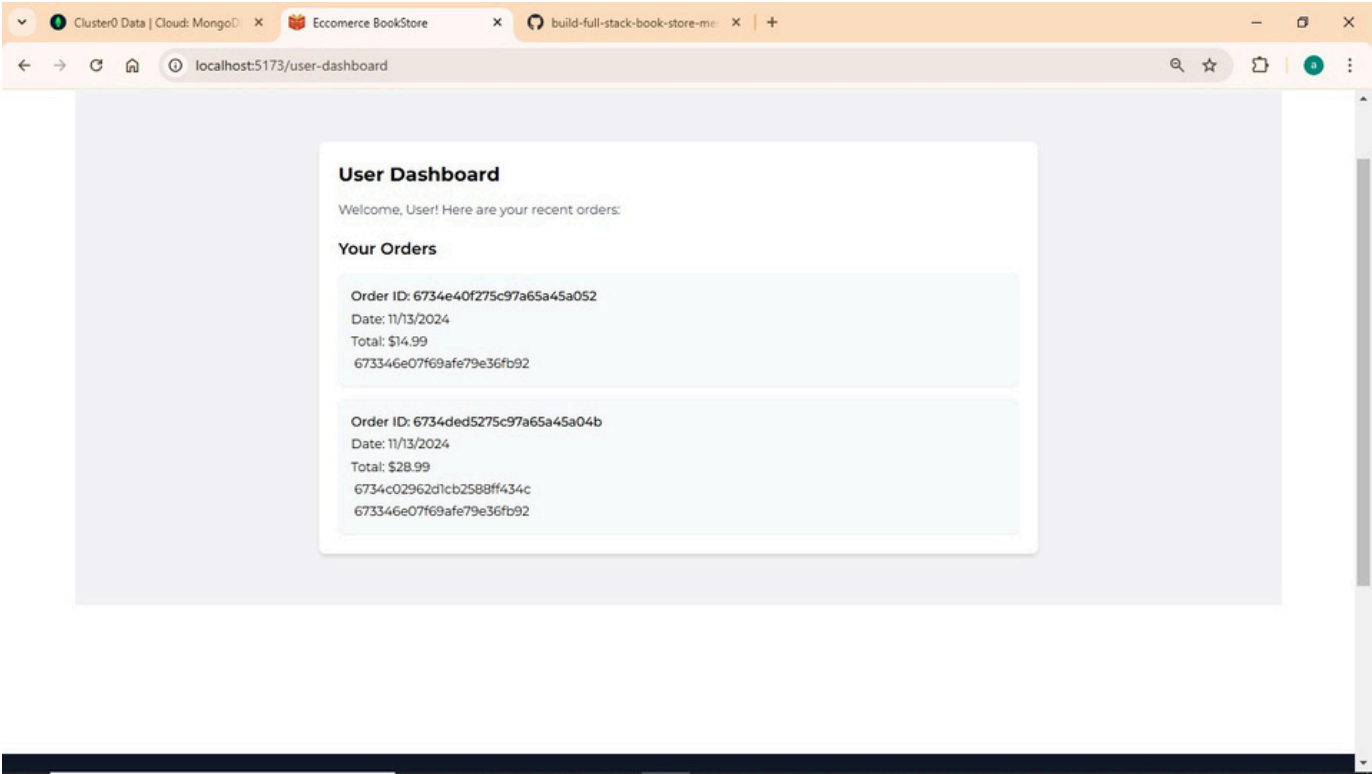
Update Book:

The screenshot shows a web browser window with the URL `localhost:5173/dashboard/edit-book/6734c02962d1cb258ff434c`. The page layout is identical to the 'Add New Book' page. The central form is titled 'Update Book' and contains the following fields:

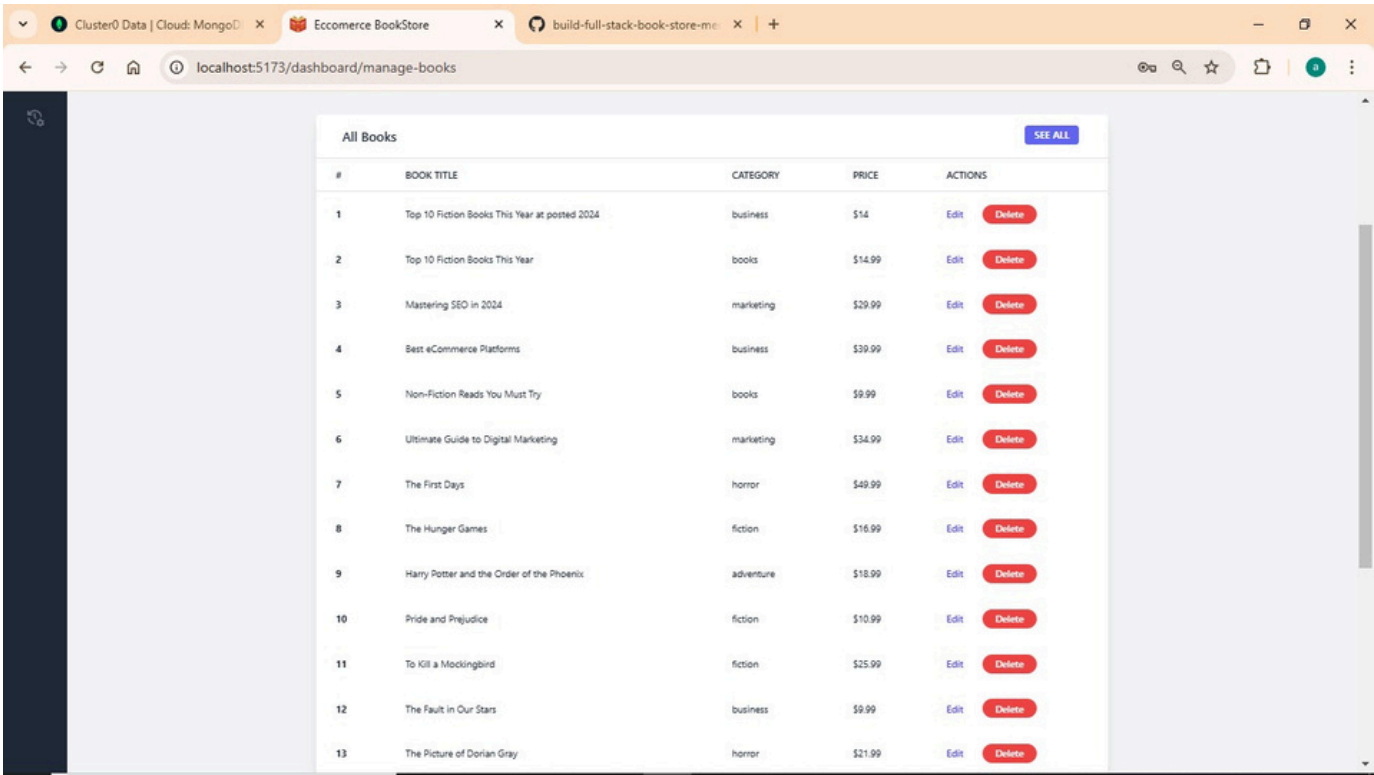
- Title:** A text input field containing 'Top 10 Fiction Books This Year at posted 2024'.
- Description:** A text input field containing 'A curated list of the best fiction books that are trending year.'
- Category:** A dropdown menu with the selected option 'Business'.
- Trending:** A checked checkbox.
- Old Price:** A text input field containing '24'.
- New Price:** A text input field containing '14'.
- Cover Image URL:** A text input field containing 'book-2.png'.

At the bottom of the form is a blue button labeled 'Update Book'.







User’s Dashborad:



All Book:



All Users:

Recent Transactions					<input type="text" value="Search"/>		 FILTER
NAME	EMAIL ID	LOCATION	AMOUNT	TRANSACTION DATE			
 Alex	alex@dashwind.com	Paris	\$100	9 May			
 Ereena	ereena@dashwind.com	London	\$190	8 May			
 John	jhon@dashwind.com	Canada	\$112	8 May			
 Matrix	matrix@dashwind.com	Peru	\$111	8 May			
 Virat	virat@dashwind.com	London	\$190	7 May			
