Full Stack Development with MERN - Project Documentation

Introduction:

The Book Store application is a full-stack web application built using the MERN stack, which includes **MongoDB**, **Express.js**, **React.js**, and **Node.js**. This application provides an intuitive and user-friendly platform for users to explore, purchase, and manage a wide selection of books online.

Book-Store Application

Team Members and Roles:

Archana A - Frontend Developer:

- Designs and develops the React-based frontend user interface, ensuring a smooth user experience.

Jeyalakshmi Subiksha S - Backend Developer:

- Builds the server-side logic, API endpoints, and handles the integration with MongoDB.

Keerthana B - Database Administrator:

- Manages the MongoDB database, optimizes queries, and ensures data integrity.

Myvizhi P - Project Manager:

- Coordinates tasks among team members and oversees project progress.

Project Description:

Welcome to the literary haven of the digital age—introducing our revolutionary Book-tore 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.

Goals:

1. Create a seamless shopping experience for users.

2. Enable sellers to manage book listings efficiently.

3. Allow admins to maintain platform integrity and monitor activity.

Scenario Based Case Study:

Sarah is an avid reader with a passion for exploring new genres and authors. However, her busy schedule often leaves her with limited time to visit physical bookstores. Sarah is looking for a solution that allows her to discover and purchase books conveniently, without compromising her reading preferences or the joy of browsing through a bookstore.

- **User Registration and Authentication:**

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

- **Book Listings:**

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

- **Book Selection:**

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

- **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.

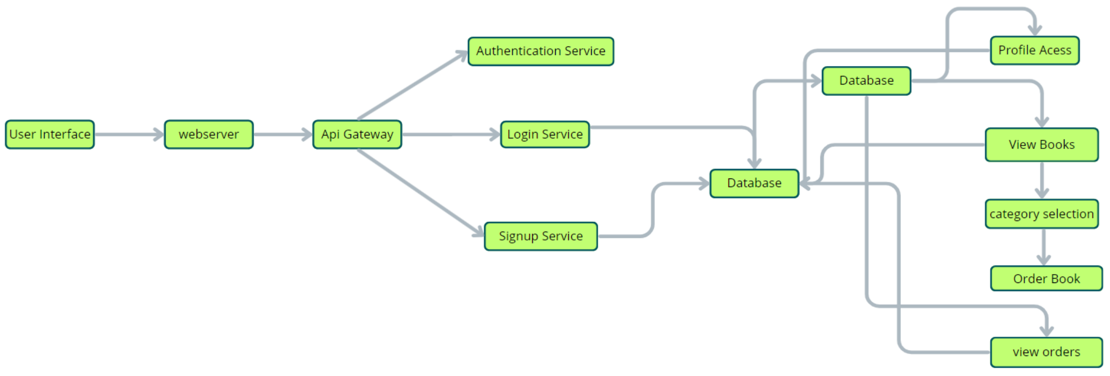
- **Order Confirmation:**

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

- **Order History:**

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

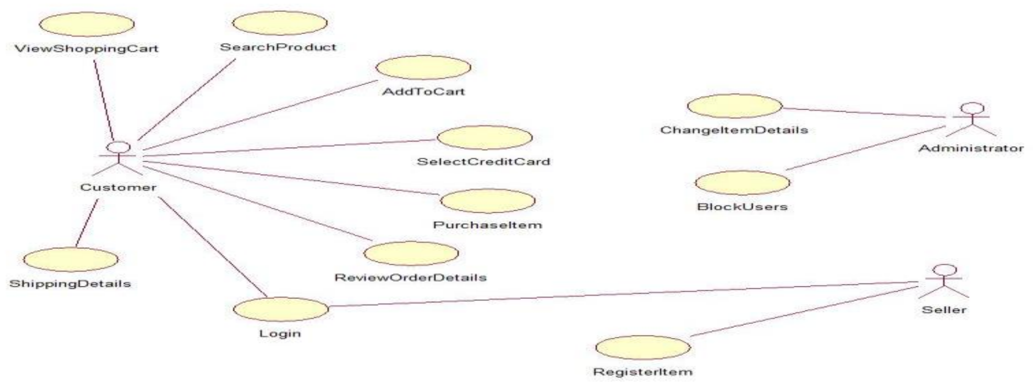
Technical Architecture:



- **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.
- **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.

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

- **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 rating
s.

Book-Inventory Relationship:

- **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:

- **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.

Additional Relationships:

- **Book-Author Relationship: Many-to-Many (M:M):** A book can have multiple authors, and an author can write multiple books. (Similar to User-Book, use an intermediate "WrittenBy" table)
- **Book-Genre Relationship: Many-to-Many (M:M):** A book can belong to multiple genres, and a genre can have many books. (Similar to User-Book, use an intermediate "CategorizedAs" table)
- **Review-User Relationship: Many-to-One (M:1):** A review is written by one user, but a user can write many reviews. (Keep UserID as a foreign key in the Review table)

Key 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.

7. **Organizer Dashboard:**
   Offer administrators an interface to manage book listings, inventory levels, user accounts, orders, and other platform-related activities.

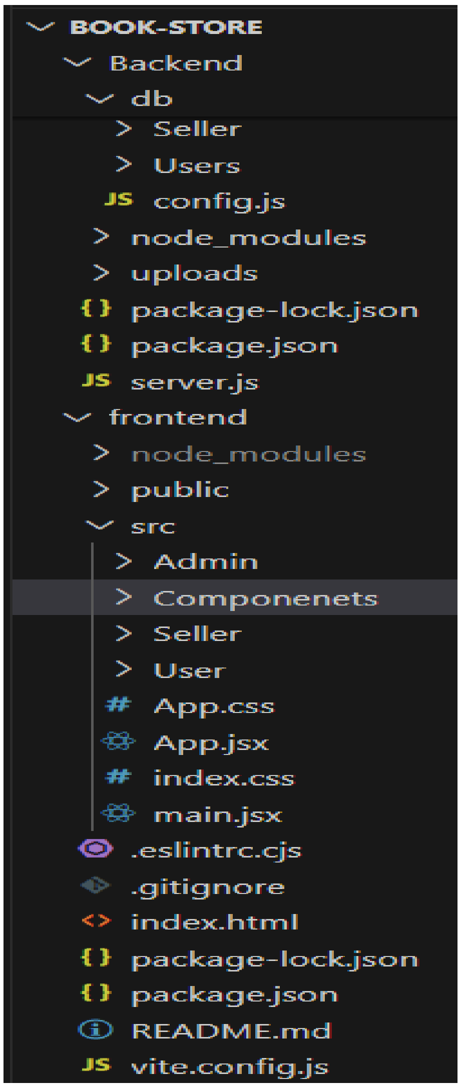
8. **Create Item:**
   Organizer can create items and add new items and he can get the items and he can update items.

9. **Admin Dashboard:**
   Offer administrators an interface to manage book listings, inventory levels, user accounts, orders, and other platform-related activities. Manage the users and organizers.

10. **Reporting and Analytics:**
    Generate reports and analytics on book sales, popular genres, user demographics, and other relevant metrics to gain insights into platform usage and performance.

11. **Integration with External APIs:**
    Integrate with third-party APIs for services like payment processing, shipping logistics, and book recommendations to enhance the functionality and user experience of the book store platform.

Project Structure:



```
BOOK-STORE
  Backend
    db
      Seller
      Users
      config.js
    node_modules
    uploads
    package-lock.json
    package.json
    server.js
  frontend
    node_modules
    public
    src
      Admin
      Componenets
      Seller
      User
      App.css
      App.jsx
      index.css
      main.jsx
    .eslintrc.cjs
    .gitignore
    index.html
    package-lock.json
    package.json
    README.md
    vite.config.js
```

Folder Structure:

Frontend Structure:

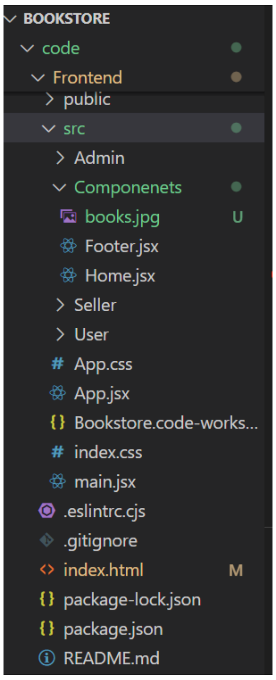
src: Main application code.

Admin, User, Seller: Organized by user role.

Components: Shared components.

App.jsx: Root component managing layout and routing.

main.jsx: Entry point for rendering.



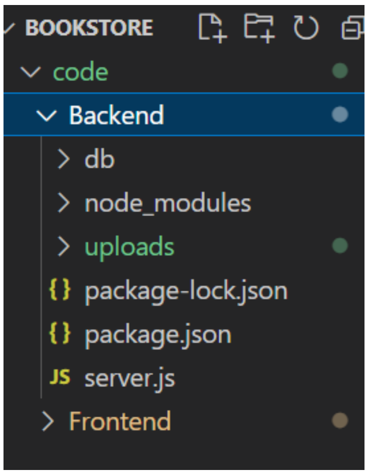
```
BOOKSTORE
  code
    Frontend
      public
      src
        Admin
        Componenets
          books.jpg          U
          Footer.jsx
          Home.jsx
        Seller
        User
        App.css
        App.jsx
        Bookstore.code-works...
        index.css
        main.jsx
      .eslintrc.cjs
      .gitignore
      index.html             M
      package-lock.json
      package.json
      README.md
```

Backend Structure:

server.js: Configures API routes and middlewares.

db/config.js: Database connection setup.

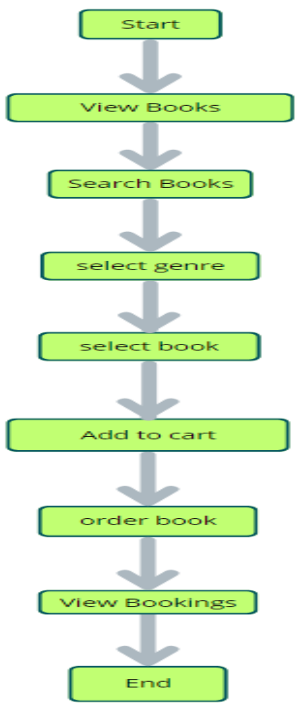
uploads: Stores any uploaded files.



```
BOOKSTORE
  code
    Backend
      db
      node_modules
      uploads
      package-lock.json
      package.json
      server.js
    Frontend
```

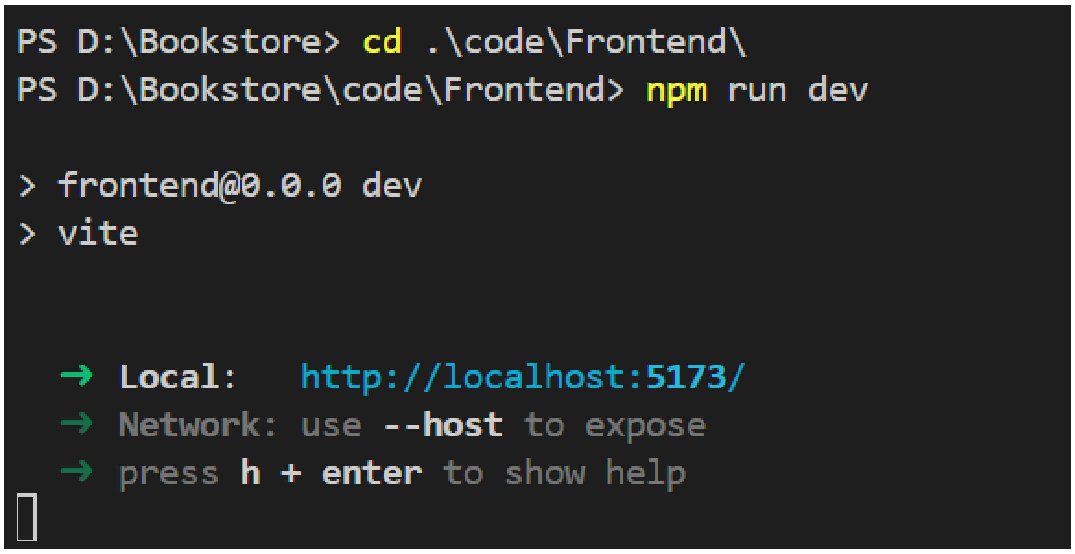
Application Flow:

- **Start:** Users open the BookEase app to explore a vast collection of books.
- **Home Page:** Users land on the home page, which provides an overview of the book store's offerings. From here, they can navigate to various sections of the app.
- **Access Profile:** Users have the option to access their profiles, allowing them to view or update personal information, preferences, and order history.
- **Book Selection:** After accessing their profiles, users proceed to browse and select books to purchase. The app presents a list of available books, along with details such as title, author, genre, and price.
- **Book Purchase**: Users navigate through the available book options and specify the quantity of each book they wish to purchase. They can also choose additional options such as e-book format or special editions.
- **View Orders:** Users have the option to view their current and past orders. This section provides details about ordered books, order status, and payment history.
- **Order Confirmation:** For new purchases, users can initiate the ordering process. This involves selecting books, specifying quantities, confirming the order, and receiving an order confirmation.
- **End:** The flow concludes as users have completed their desired actions within the BookEase app



Running The Application:

Frontend: Start React server with:

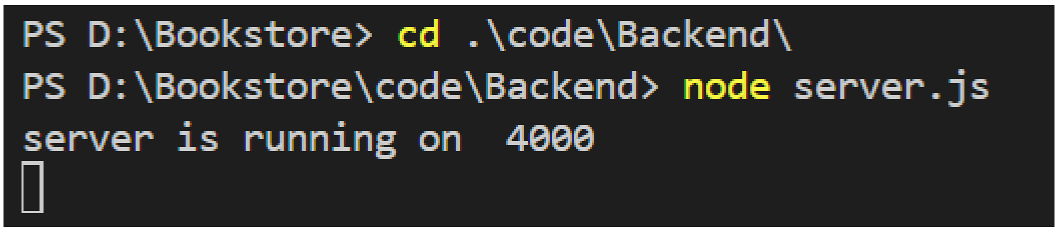


```
PS D:\Bookstore> cd .\code\Frontend\
PS D:\Bookstore\code\Frontend> npm run dev

> frontend@0.0.0 dev
> vite


  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

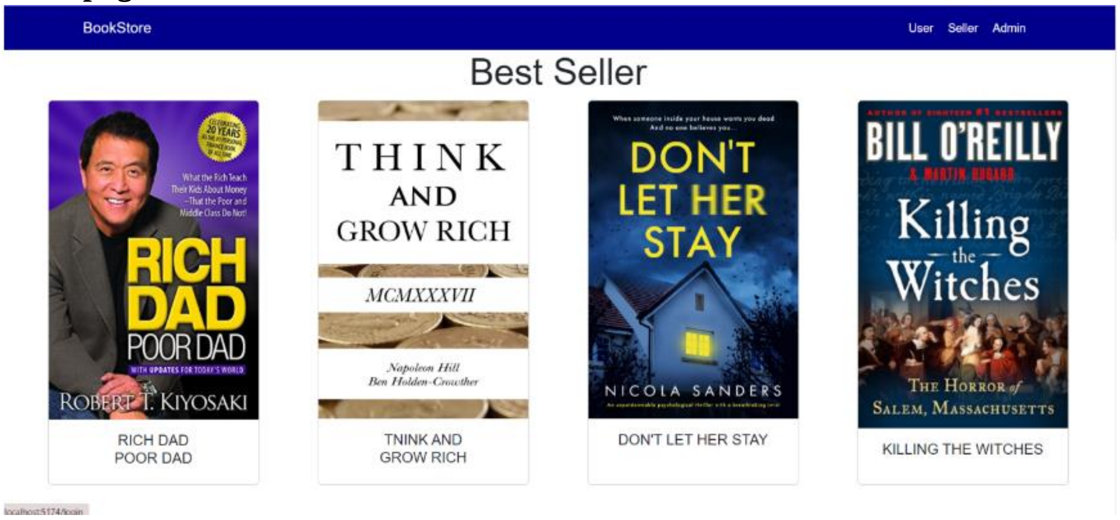
Backend: Start Express server with:



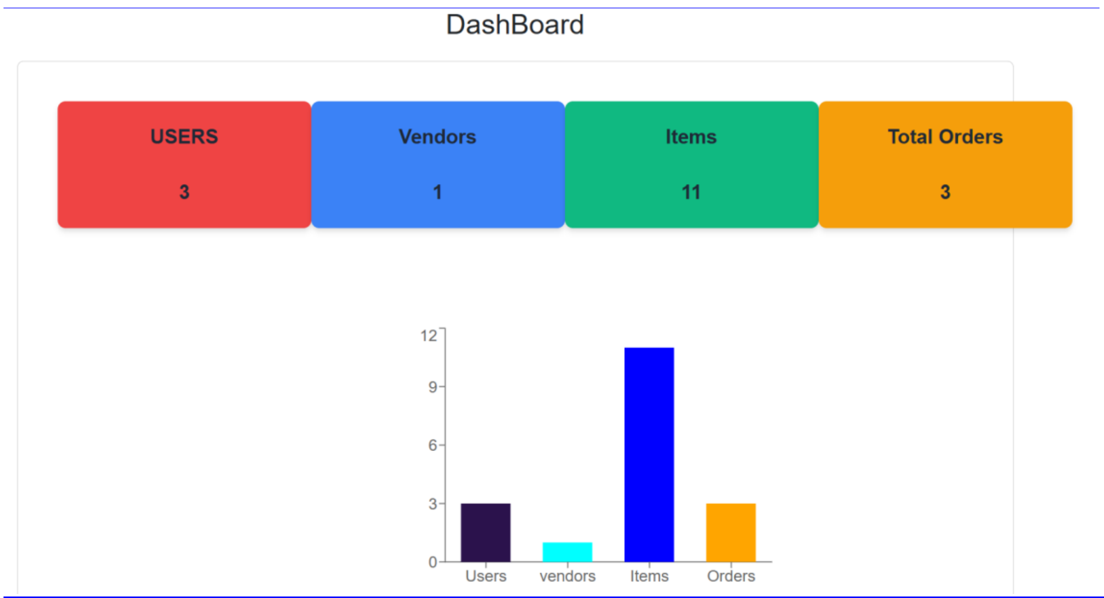
```
PS D:\Bookstore> cd .\code\Backend\
PS D:\Bookstore\code\Backend> node server.js
server is running on  4000
```

User Interface:

Homepage:

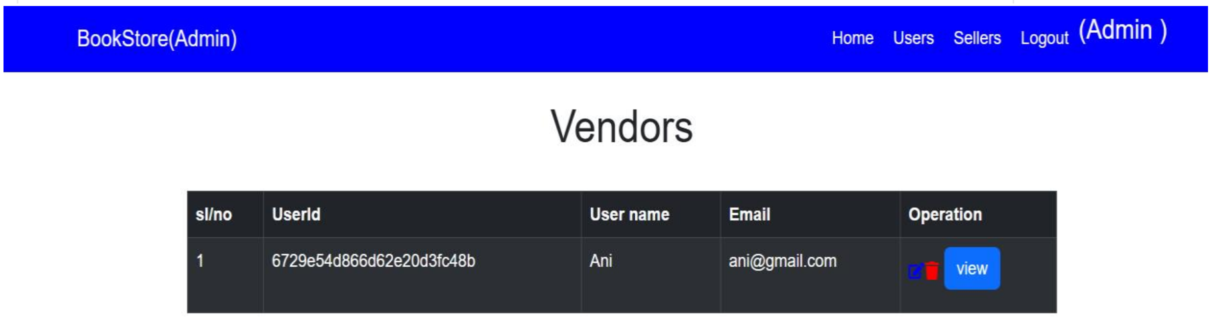


Admin Dashboard:



DashBoard

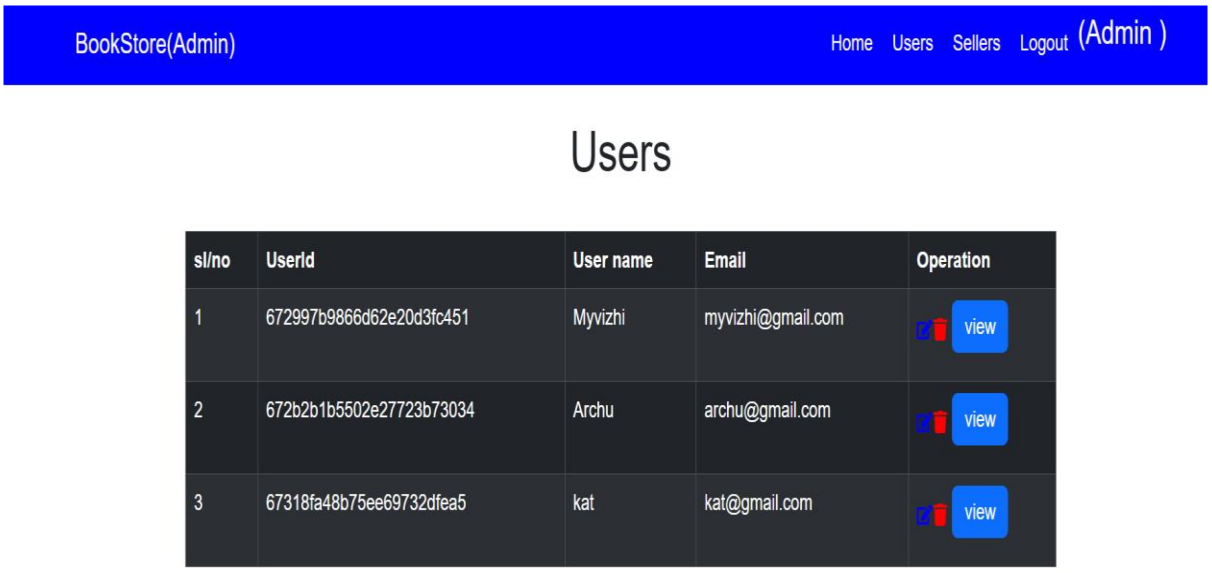
USERS	Vendors	Items	Total Orders
3	1	11	3



BookStore(Admin) Home Users Sellers Logout (Admin)

Vendors

sl/no	UserId	User name	Email	Operation
1	6729e54d866d62e20d3fc48b	Ani	ani@gmail.com	view

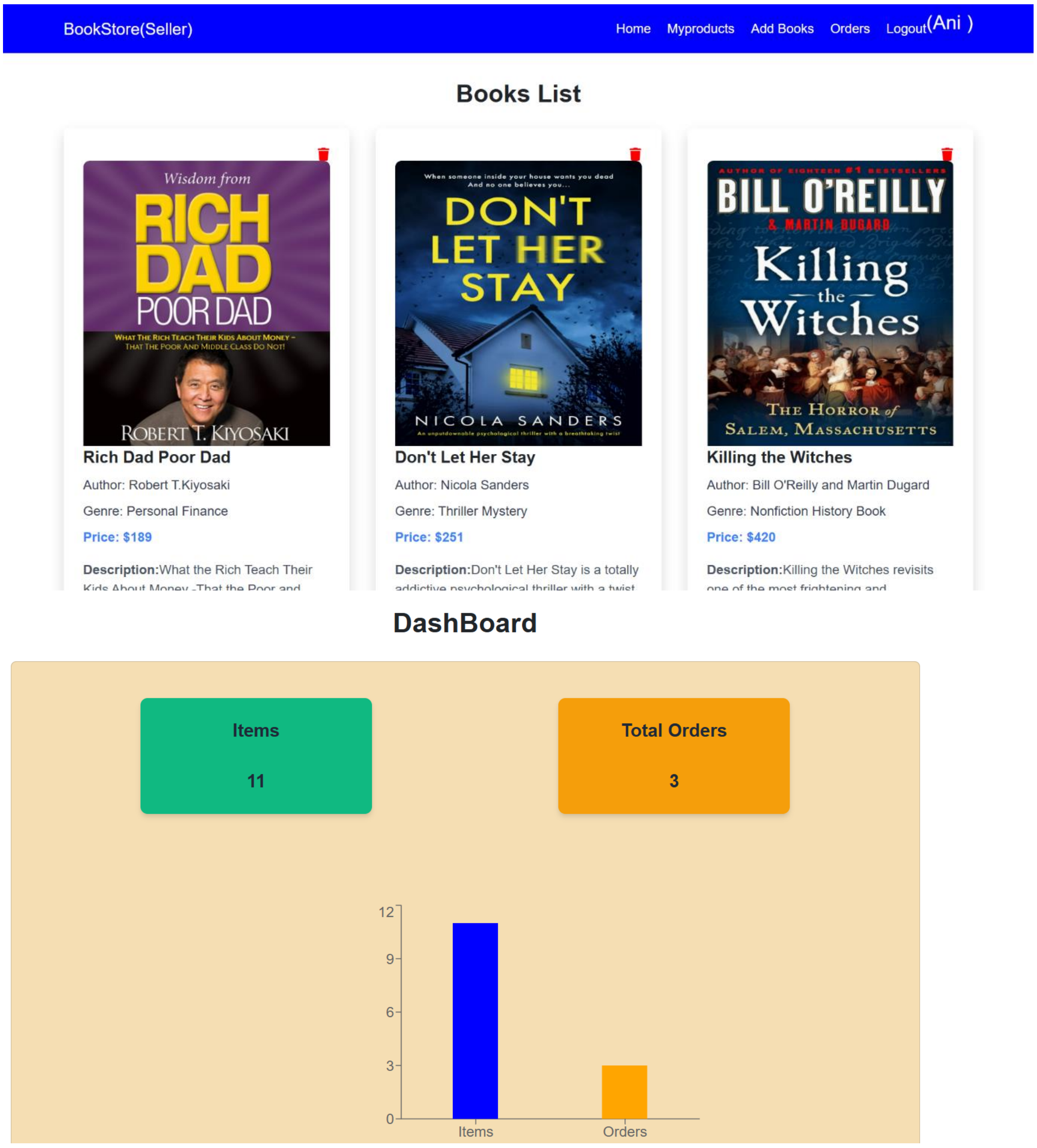


BookStore(Admin) Home Users Sellers Logout (Admin)

Users

sl/no	UserId	User name	Email	Operation
1	672997b9866d62e20d3fc451	Myvizhi	myvizhi@gmail.com	view
2	672b2b1b5502e27723b73034	Archu	archu@gmail.com	view
3	67318fa48b75ee69732dfea5	kat	kat@gmail.com	view

Seller Dashboard:


BookStore(Seller)

Home Myproducts Add Books Orders Logout(Ani)

Books List

Rich Dad Poor Dad

Author: Robert T.Kiyosaki

Genre: Personal Finance

Price: $189

Description: What the Rich Teach Their Kids About Money -That the Poor and

Don't Let Her Stay

Author: Nicola Sanders

Genre: Thriller Mystery

Price: $251

Description: Don't Let Her Stay is a totally addictive psychological thriller with a twist

Killing the Witches

Author: Bill O'Reilly and Martin Dugard

Genre: Nonfiction History Book

Price: $420

Description: Killing the Witches revisits one of the most frightening and

DashBoard

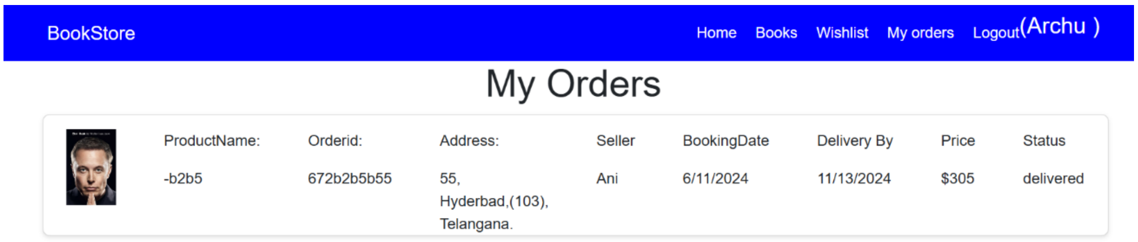
Items

11

Total Orders

3

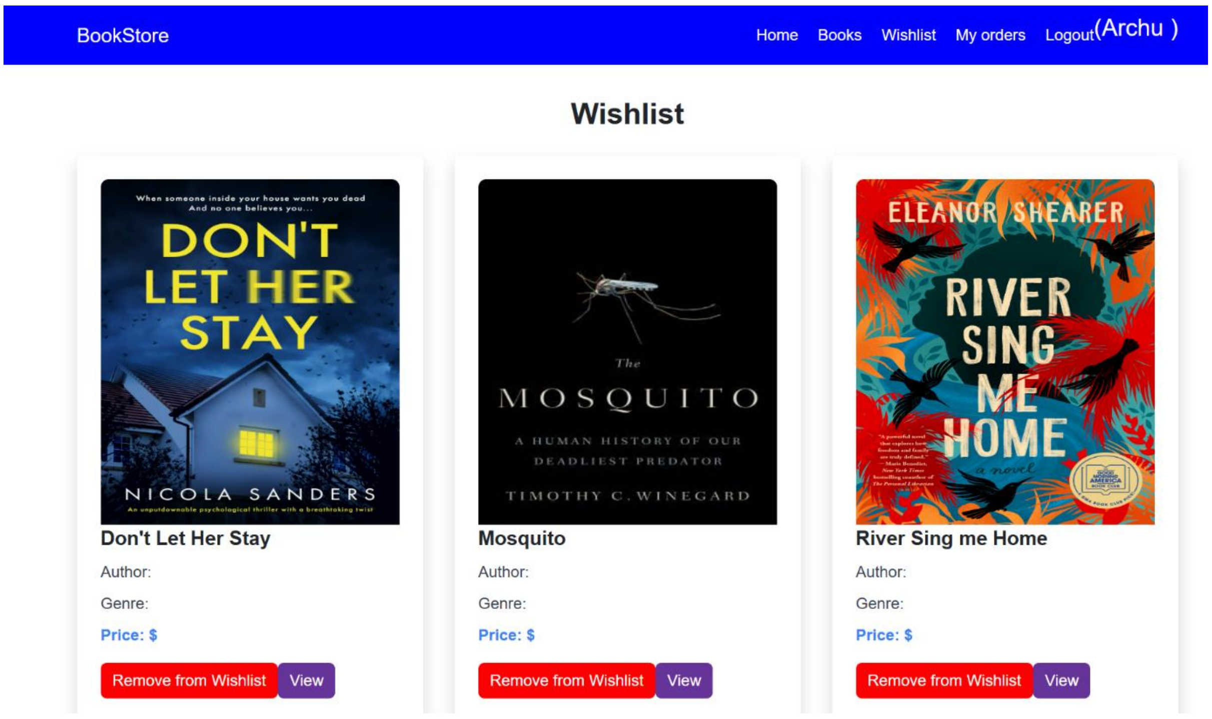
User Profile:



BookStore | Home | Books | Wishlist | My orders | Logout(Archu)

My Orders

	ProductName:	Orderid:	Address:	Seller	BookingDate	Delivery By	Price	Status
	-b2b5	672b2b5b55	55, Hyderbad,(103), Telangana.	Ani	6/11/2024	11/13/2024	$305	delivered



BookStore | Home | Books | Wishlist | My orders | Logout(Archu)

Wishlist

Don't Let Her Stay
Author:
Genre:
Price: $

Mosquito
Author:
Genre:
Price: $

River Sing me Home
Author:
Genre:
Price: $

Demo / Github link:

Known Issues:

1. Token Expiry:

Issue:

After prolonged sessions, the user's authentication token may expire, requiring them to log in again. This can be inconvenient, especially for users who spend extended time browsing or leave their sessions open in the background.

Solution:

A potential fix would be to implement a refresh token mechanism, which can periodically renew the user's session without requiring re-authentication. Alternatively, setting a longer expiry time for the token or providing a "Remember Me" option on login could improve the user experience. Adding feedback or a notification upon expiry would also help inform users.

2. Database Delays:

Issue:

The application may experience latency when querying large datasets, particularly as the catalog of books grows. This could result in slower load times when displaying books or searching through extensive lists.

Solution:

Indexing the MongoDB collections and optimizing query patterns can help reduce load times. Additionally, implementing pagination and caching frequently accessed data (such as popular books) could further enhance response times. For high-traffic scenarios, considering database sharding or clustering may be beneficial.

Future Enhancements:

1. Wishlist Functionality:

Description:

The wishlist feature will allow users to save books that they are interested in but not ready to purchase immediately. This would let them easily revisit and purchase these books later.

Benefits:

The wishlist provides a personalized user experience and can increase user engagement by encouraging users to return to their saved items. It may also boost sales by helping users remember books they want to buy in the future.

Implementation:

This feature could involve adding a "Save to Wishlist" button on each book's page. Saved books could be stored in the user's profile in the database, making them accessible across sessions. Notifications or reminders could also be added to prompt users about their saved books.

2. User Reviews

Description:

Allowing users to rate and review books enhances the community aspect of the application. Users could leave star ratings, write reviews, and read others' feedback before making purchasing decisions.

Benefits:

User reviews can improve transparency and trust, as customers gain insights from the experiences of others. Positive reviews may drive sales, while constructive feedback helps in guiding future customers.

Implementation:

This feature could include a review form on each book's page, with ratings and comments stored in the database and linked to both the user and the book. Displaying average ratings and the number of reviews could provide quick insights for prospective buyers. The backend can support CRUD operations to allow users to add, edit, and delete their reviews.