BOOK EASE

Naan Mudhalvan - Project Documentation

Introduction

Project Title: book ease

College: Tagore Engineering College – 4127

Department: B.E CSE IV Year 7th Sem

Team ID: NM2024TMID02189

Team Members:

S.No	Name	Register Number	Nm Id
1	Nithishkumar S	412721104032	8c666c4851e544bbb904951222b8320b
2	Roshini D	412721104040	97d6993d962881c97784a001b70e1e40
3	Kiruthika E	412721104023	496fa98e0e738e8a53d4d2620194b834
4	Sagaya Raj D	412721104041	Bf3c1f95193518abaf141aecb3f8f0ff
5	Sumathi V	412721104051	8057b8a54c2a735ada4430f29d16db96

Project Overview

Purpose:

BookEase is an application that allows users to explore a vast collection of books, make purchases, and manage their profiles with ease. This documentation provides an overview of the app's features and user flow.

Features:

- **Vast Collection of Books**

Explore a wide variety of books across different genres.

- **User Profiles**

Users can create and manage their profiles, including personal information and order history.

- **Book Selection**

Browse available books with details such as title, author, genre, and price.

- **Book Purchase**

Easily purchase books by selecting quantities and options like e-book formats or special editions.

- **Order Management**

View current and past orders, including order status and Details.

- **Order Confirmation**

Receive confirmation for new purchases after selecting books and quantities.

Architecture

- **Frontend:**

Developed using React, styled with Tailwind, and built with Vite for optimized performance.

Responsive design ensures usability across devices.

- **Backend:**

Built using Express.js to handle APIs and business logic.

Middleware includes:

- bcryptjs for password hashing.
- jsonwebtoken for authentication.
- Multer for file handling.

- **Database:**

MongoDB as the database solution, implemented with Mongoose for schema design and data interaction

Setup Instructions

Prerequisites:

- Node.js
- MongoDB Atlas Account
- Cloudinary account for image storage

Installation:

1. **Clone the repository:**

git clone https://github.com/Nithishkumar2004/Book-store.git

2. **Install dependencies:**

- **Frontend:**

cd frontend

npm install

- **Backend:**

  cd backend

  npm install

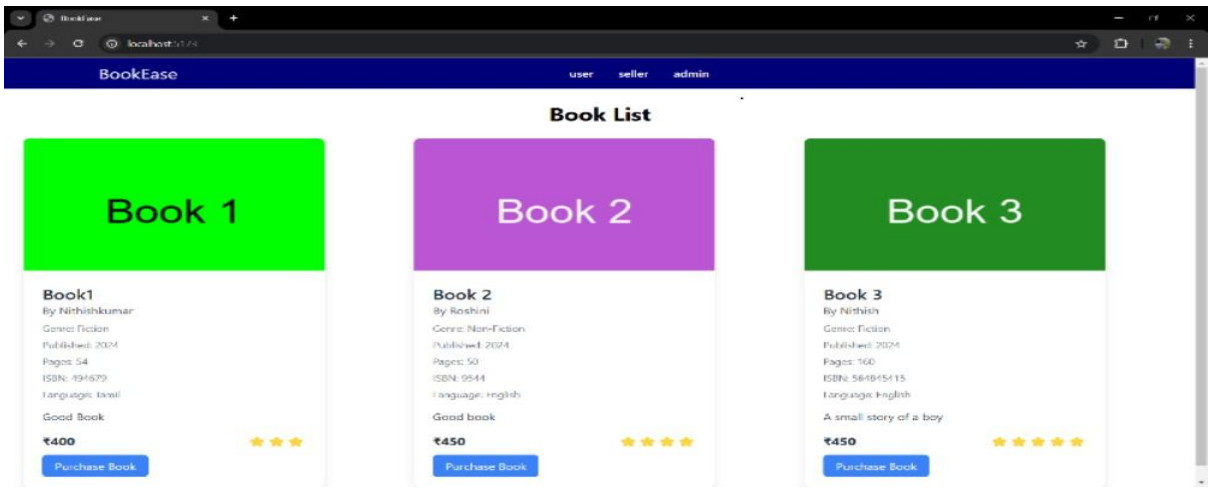
3. Set up environment variables:

- Create .env file in the server folder with the following:

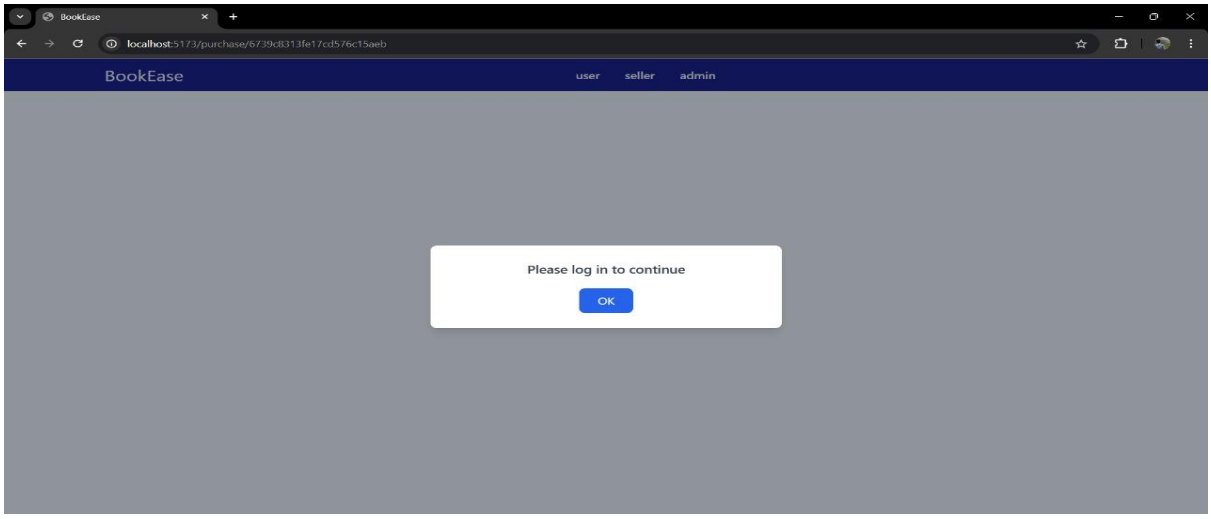
```
PORT=3000
mongoDBURL='mongodb+srv://nithishcse2021:qAcv0YyZNkP18MgO@cluster0.z7clm.mongodb.net/Bookstore?retryWrites=true&w=majority&appName=Cluster0'
REACT_APP_ENDPOINT='http://localhost:3000'
JWT_SECRET='b6188365be95d62f6dc082f1844e4f3eb862bc8f41438bf5c2c13d2aaa7776e0b251af76795211f34779860d5d5a2c3288590e0921695d51510c13c814a37830fded89ecbc9941f3c16e41e90fd889f1e009a8d31352325947eb8cf43db7905626027664b586b9dc3586a0873645232770821185a2833cdd350a31701c5944764734e4b6cf1daf4b05286664c8f44c45c90c0d24db43292229b3d20807d43c3dacc7d608877ede662d15f16b9406b3e5eb22a3a5574df6a39900a9df4141831b0a3ad42ee9431db12c0236777d9c37ca4c0f732446ca898496ca94360eadc8d065dd5fb7792111ed7c20137c7112698c11928d253819989fa4e77750156b8ff7'
JWT_EXPIRY=3600000  # Optional: Set the expiry time for JWT
CLOUDINARY_CLOUD_NAME='dvbgs8fy1'
CLOUDINARY_API_KEY='148237359281976'
CLOUDINARY_API_SECRET='HN_KnZyL-oaKeyBSlF3BVoEWbOU'
```

Screenshots or Demo

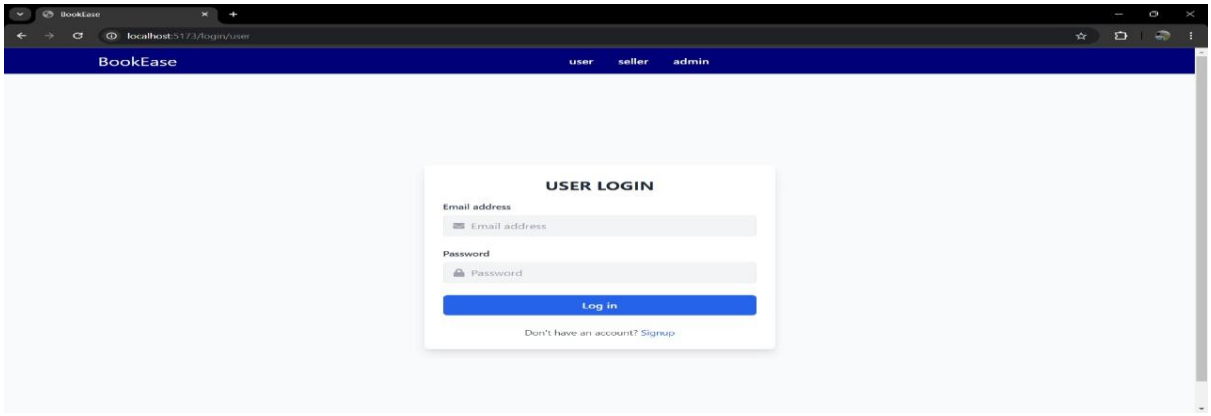
1.landing page:



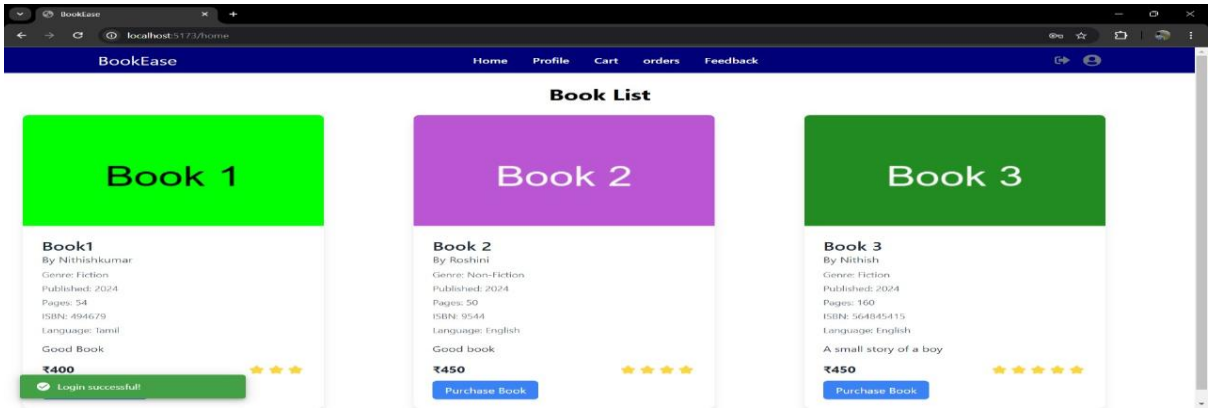
2. Authentication for purchasing book



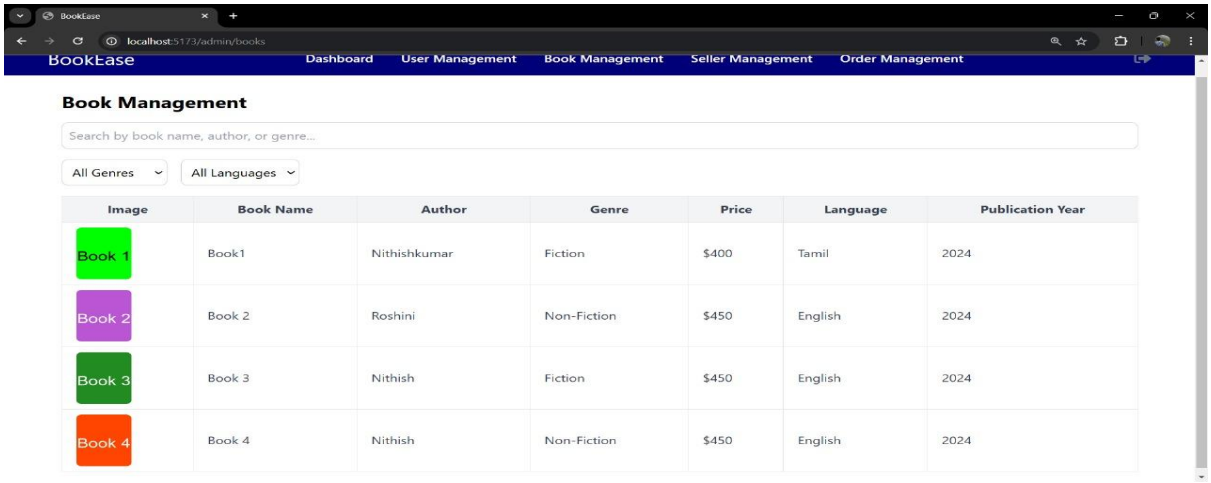
3.user login



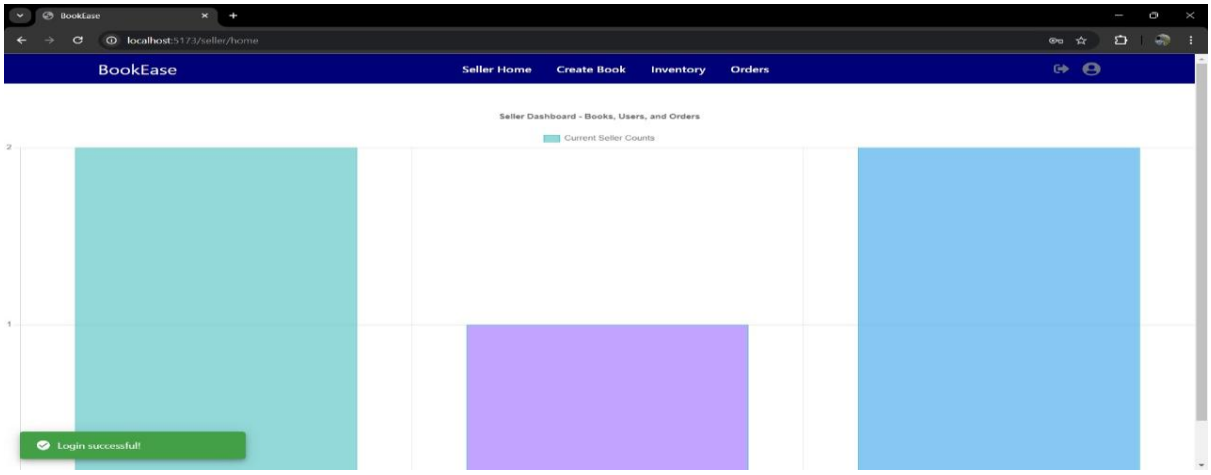
4.user dashboard



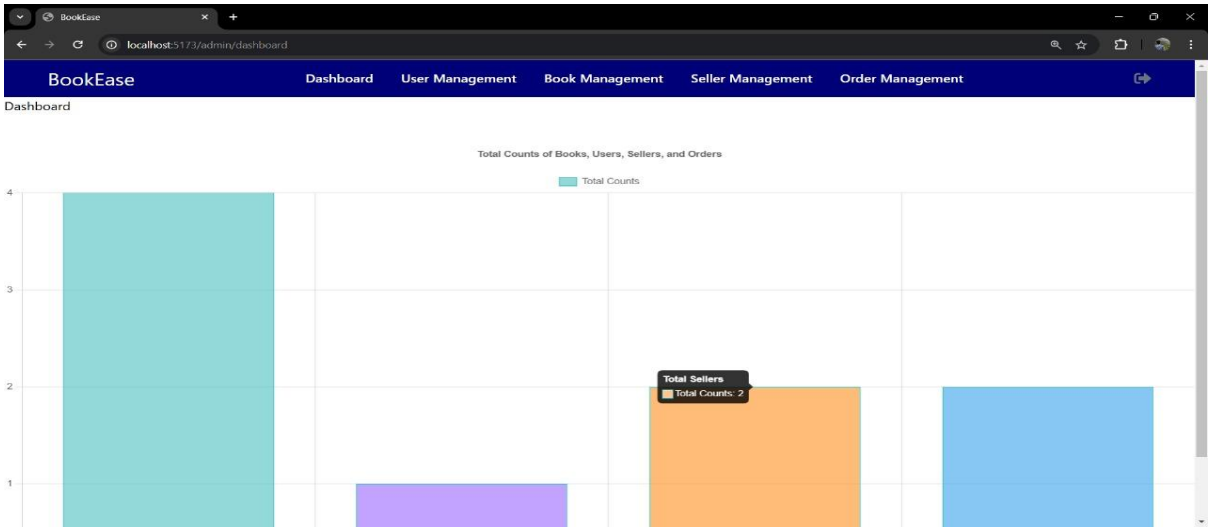
5.book page



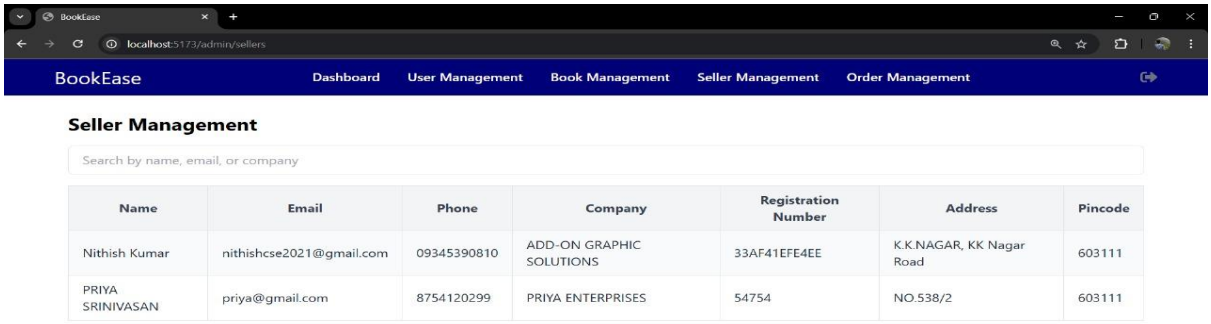
6.seller dashboard



7.admin dashboard:

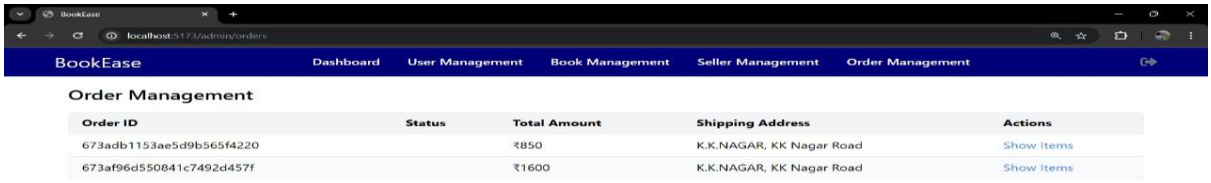


8.seller page



Name	Email	Phone	Company	Registration Number	Address	Pincode
Nithish Kumar	nithishcse2021@gmail.com	09345390810	ADD-ON GRAPHIC SOLUTIONS	33AF41EFE4EE	K.K.NAGAR, KK Nagar Road	603111
PRIYA SRINIVASAN	priya@gmail.com	8754120299	PRIYA ENTERPRISES	54754	NO.538/2	603111

9.order page



Order ID	Status	Total Amount	Shipping Address	Actions
673adb1153ae5d9b565f4220		₹850	K.K.NAGAR, KK Nagar Road	Show Items
673af96d550841c7492d457f		₹1600	K.K.NAGAR, KK Nagar Road	Show Items

Known Issues:

- Limited Error Handling:
  The system lacks robust error handling for invalid book uploads.
- No Payment Integration:
  Payment gateway is not implemented as per project requirements.

Future Enhancements:

- Add Payment Gateway:
  Integrate a secure payment gateway for purchasing books.
- Recommendation Engine:
  Develop a personalized recommendation system to suggest books based on user interests and purchase history.
- E-Reader Integration:
  Include an integrated e-reader for seamless access to purchased e-books within the platform.

Route	HTTP Method	Type	Purpose
/orders	GET	Admin	Fetch all orders to provide an overview of transactions or sales.
/counts	GET	Admin	Retrieve book count for users, sellers, and the total book count.
/login	POST	Admin	Authenticate admin login using provided credentials.
/sellers	GET	Admin	Fetch a list of all sellers registered on the platform.
/users	GET	Admin	Fetch a list of all users registered on the platform.
/books	GET	Admin	Fetch details of all books available in the system.
/uploadimage	POST	Book	Upload a single image for a book (e.g., book cover) to the server.
/create	POST	Book	Create a new book entry in the database.
/book/:id	GET	Book	Fetch details of a specific book using its unique ID.
/books	GET	Book	Retrieve details of all books available in the database.
/create	POST	Order	Create a new order and save it to the database.
/update/:orderId	PATCH	Order	Update the status or attributes of an order (e.g., mark as shipped) and adjust inventory.
/update-inventory/:bookId	PATCH	Order	Update the inventory count for a specific book based on stock changes.
/cancel/:orderId	PATCH	Order	Cancel an order and make necessary updates (e.g., refund or inventory reset).
/orders	GET	Order	Fetch all orders related to a specific seller.
/update-inventory/:bookId	PUT	Seller	Update inventory details for a specific book owned by the seller.

Route	HTTP Method	Type	Purpose
/counts	GET	Seller	Retrieve counts specific to the seller (e.g., total books, sales, etc.).
/profile	PUT	Seller	Update the seller's profile information.
/inventory	GET	Seller	Fetch the seller's inventory details (list of books or stock levels).
/profile	GET	Seller	Fetch the seller's profile information.
/login	POST	Seller	Authenticate the seller's login credentials.
/register	POST	Seller	Register a new seller on the platform.
/addcart	POST	User	Add an item to the user's cart.
/profile	PUT	User	Update the user's profile information.
/profile	GET	User	Fetch the user's profile information.
/login	POST	User	Authenticate the user's login credentials.
/register	POST	User	Register a new user on the platform.
/cart	GET	User	Fetch the user's cart items, including book and seller details.
/cart/item/	DELETE	User	Delete a specific item from the user's cart.
/cart	DELETE	User	Clear all items from the user's cart.
/orders	GET	User	Fetch all orders placed by the user.
/cart/:id	PUT	User	Update an item in the user's cart.

Summary of the HTTP Methods:

- GET: Retrieves data (e.g., books, orders, profiles).
- POST: Creates new data (e.g., creating books, orders, users).
- PUT: Updates existing data (e.g., profiles, cart items).
- PATCH: Updates specific parts of data (e.g., order status, inventory).
- DELETE: Removes data (e.g., items from cart).

This table now includes Admin, Seller, Book, Order, and User routes with all their respective methods and purposes.