**Full Stack Development with MERN**

**Project Documentation format**

1. **. Introduction** 
   1. **Project Title:** Book Store - Website Name : BookNest
   2. **Team Members:**
   3. **G. Aakash**
   4. **G. Harshith**
   5. **J. Marthand Bhargav**
   6. **V . Narendra**

1. **. Project Overview**

**Purpose:**

The purpose of the BookSell Store project is to provide a platform where users can sell their  second-hand books and purchase new or used books. The platform also integrates with the Google  Play Book API for book searches and Razorpay for payment processing.

**Features:**

* 1. User authentication and authorization.
  2. Sell second-hand books with image uploads.
  3. Search and get books info which is fetched through the Google Play Book API.
  4. Manage books, orders, and users in the admin dashboard.

1. **Architecture**

**Frontend:**

The frontend is built using React.js and Bootstrap for responsive UI design. React Router is used for  routing, and Axios is used for making HTTP requests.

**Backend:**

The backend is developed using Node.js and Express.js. It handles user authentication, book  listings, order processing, and integration with third-party APIs.

**Database:**

MongoDB is used as the database, managed using MongoDB Compass. The database schema  includes collections for users, books, orders .

1. **Setup Instructions**

**Prerequisites:**

* 1. Node.js
  2. MongoDB

**Installation:**

**Backend Setup**

* 1. . **Create the Folder**:
     1. Navigate to your desired directory and create a folder Bookstore Admin and User ○ Change your directory to the backend folder: Bookstore Admin and User /backend 2 . **Install Dependencies**:

○ Use your command line to run the command to install all necessary dependencies:

npm install

       3 . **Set Up Environment Variables**:

* ○ Create a .env file in the backend directory and MONGO\_URI: Your MongoDB connection string
* PORT: Port number for the server

4 . **Start the Backend Server**:

○ Use the command line to start the server:  node app.js

**Frontend Setup**

1. **Navigate to Frontend Directory** :

○ Change your directory to the frontend folder: Bookstore Admin and User  /frontend.

* 1. **Install Dependencies** :
     1. Use your command line to run the command to install all necessary dependencies:  npm install.
  2. **Build the Frontend** :
     1. Use the command line to build the frontend: npm run build.

1. **Folder Structure**

**Client:**

**frontend\src**

* **.dist**: Distribution files (if any).
* **components**:
  + **AdminPages**:
    - AddBook.jsx: Component for adding new books.
    - AllOrders.jsx: Component to display all orders.
    - SeeUserData.jsx: Component to view user data.
    - UpdateBooks.jsx: Component to update book details.
  + **Books**:
    - BookCard.jsx: Component to display individual book details.
  + **Home**:
    - Hero.jsx: Component for the main banner on the home page.
    - RecentlyAdded.jsx: Component to display recently added books.
  + **Profile**:
    - MobileBar.jsx: Component for the mobile navigation bar in the profile section.
    - Sidebar.jsx: Component for the sidebar in the profile section.
  + Footer.jsx: Component for the footer of the website.
  + Navbar.jsx: Component for the navbar of the website.
* **pages**:
  + AllBooks.jsx: Page to display all books.
  + Cart.jsx: Page for the shopping cart.
  + Favourite.jsx: Page to display favorite books.
  + Home.jsx: Home page.
  + Loader.jsx: Page or component for loading animations.
  + Login.jsx: Login page.
  + OrderHistory.jsx: Page to display order history.
  + Profile.jsx: Profile page.
  + Settings.jsx: Settings page.
  + Signup.jsx: Signup page.
  + ViewBookDetails.jsx: Page to view detailed information about a book.
* **store**:
  + auth.js: Authentication-related state management.
  + index.js: Main store configuration.
* App.css: Global styles for the application.
* App.jsx: Main application component and routing.
* index.css: Global styles.
* main.jsx: Entry point for the application.

**Server:**

**backend**

* **.dist**: Distribution files (if any).
* **conn**:
  + conn.js: Database connection configuration.
* **models**:
  + book.js: Schema and model for books.
  + order.js: Schema and model for orders.
  + user.js: Schema and model for users.
* **routes**:
  + **.dist**: Distribution files for routes (if any).
  + book.js: Routes for book-related operations.
  + cart.js: Routes for cart-related operations.
  + favourite.js: Routes for favorite books operations.
  + order.js: Routes for order-related operations.
  + user.js: Routes for user-related operations.
  + userAuth.js: Routes for user authentication.
* **.env**: Environment variables configuration file.
* app.js: Main application file.

**6 . Running the Application  Frontend:**

1. cd frontend - npm run dev

**Backend:**

1. cd backend – node app.js
2. **API Documentation  Endpoints:**

**7. API Documentation  Endpoints:**

**GET /books**

Retrieves book data for display from the database.

* **Response**:
* {
* "url": { "type": "String", "required": true },
* "title": { "type": "String", "required": true },
* "author": { "type": "String", "required": true },
* "price": { "type": "Number", "required": true },
* "desc": { "type": "String", "required": true },
* "language": { "type": "String", "required": true }
* }

**POST /books**

Adds a new book to the database.

* **Request**:
* {
* "url": { "type": "String", "required": true },
* "title": { "type": "String", "required": true },
* "author": { "type": "String", "required": true },
* "price": { "type": "Number", "required": true },
* "desc": { "type": "String", "required": true },
* "language": { "type": "String", "required": true }
* }

**POST /login**

Authenticates a user.

* **Request**:
* {
* "email": { "type": "String", "required": true, "unique": true },
* "password": { "type": "String", "required": true }
* }

**POST /usedBooks**

Users can sell their books by uploading and it gets posted into the database.

* **Request**:
* {
* "title": { "type": "String", "required": true },
* "author": { "type": "String", "required": true },
* "category": { "type": "String", "required": true },
* "description": { "type": "String", "required": true },
* "imageUrl": { "type": "String", "required": true },
* "quantity": { "type": "Number", "required": true },
* **Response**:
* {
* "title": { "type": "String", "required": true },
* "author": { "type": "String", "required": true },
* "category": { "type": "String", "required": true },
* "description": { "type": "String", "required": true },
* "imageUrl": { "type": "String", "required": true },
* "quantity": { "type": "Number", "required": true },
* "address": { "type": "String", "required": true },

**GET /orders**

Retrieves order data for display from the database.

* **Response**:
* {
* "user": { "type": "mongoose.Types.ObjectId", "ref": "user" },
* "book": { "type": "mongoose.Types.ObjectId", "ref": "books" },
* "status": { "type": "String", "enum": ["Order placed", "Out for delivery", "Delivered", "Canceled"] }
* }

**POST /orders**

Adds a new order to the database.

* **Request**:
* {
* "user": { "type": "mongoose.Types.ObjectId", "ref": "user" },
* "book": { "type": "mongoose.Types.ObjectId", "ref": "books" },
* "status": { "type": "String", "default": "Order placed", "enum": ["Order placed", "Out for delivery", "Delivered", "Canceled"] }
* }

**GET /users**

Retrieves user data for display from the database.

* **Response**:
* {
* "username": { "type": "String", "unique": true },
* "email": { "type": "String", "required": true, "unique": true },
* "password": { "type": "String", "required": true },
* "address": { "type": "String" },
* "avatar": { "type": "String", "default": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png" },
* "role": { "type": "String", "default": "user", "enum": ["user", "admin"] },
* "favourite": [{ "type": "mongoose.Types.ObjectId", "ref": "books" }],
* "cart": [{ "type": "mongoose.Types.ObjectId", "ref": "books" }],
* "orders": [{ "type": "mongoose.Types.ObjectId", "ref": "order" }]
* }

**POST /users**

Adds a new user to the database.

* **Request**:
* {
* "username": { "type": "String", "unique": true },
* "email": { "type": "String", "required": true, "unique": true },
* "password": { "type": "String", "required": true },
* "address": { "type": "String" },
* "avatar": { "type": "String", "default": "https://cdn-icons-png.flaticon.com/128/3177/3177440.png" },
* "role": { "type": "String", "default": "user", "enum": ["user", "admin"] },
* "favourite": [{ "type": "mongoose.Types.ObjectId", "ref": "books" }],
* "cart": [{ "type": "mongoose.Types.ObjectId", "ref": "books" }],
* "orders": [{ "type": "mongoose.Types.ObjectId", "ref": "order" }]
* }

If you need more endpoints or further details, feel free to ask!

1. **. Authentication**

Authentication and authorization are handled using JSON Web Tokens (JWT). Tokens are issued  upon user login and are used to protect routes and endpoints. Sessions are managed client-side  using local storage.

1. **. User Interface**

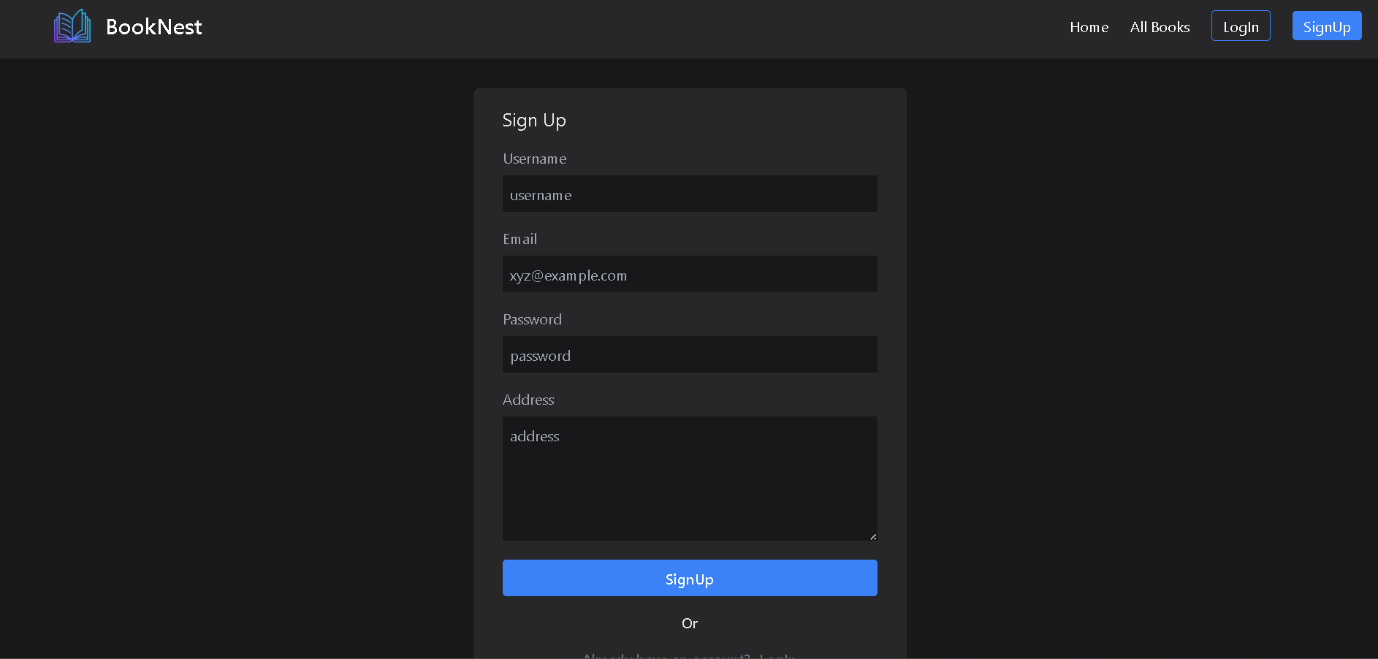
Its a simple dynamic website where the home page is driven by navbar and the page

is directed the way wise and have cards of best sellers book and used books that is to sale and the  rest is designed by simple bootstrap

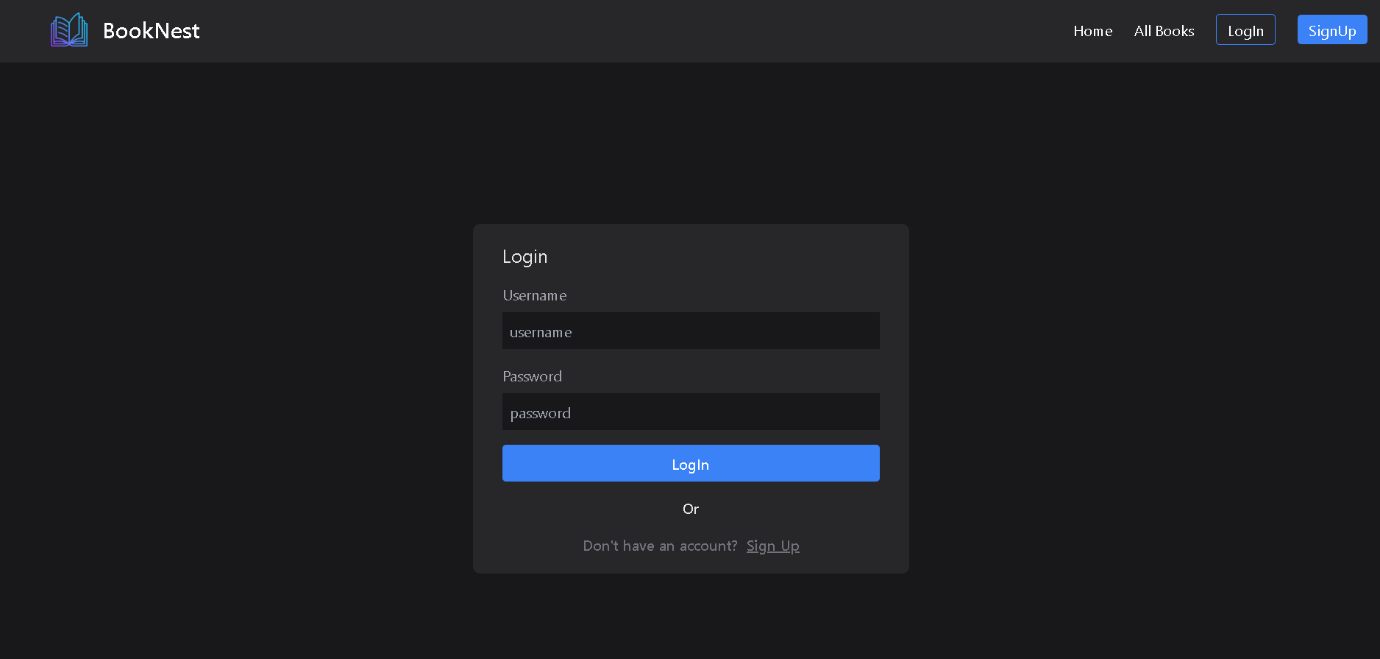
**10. Testing**

Manual test is performed and has passed all test cases.

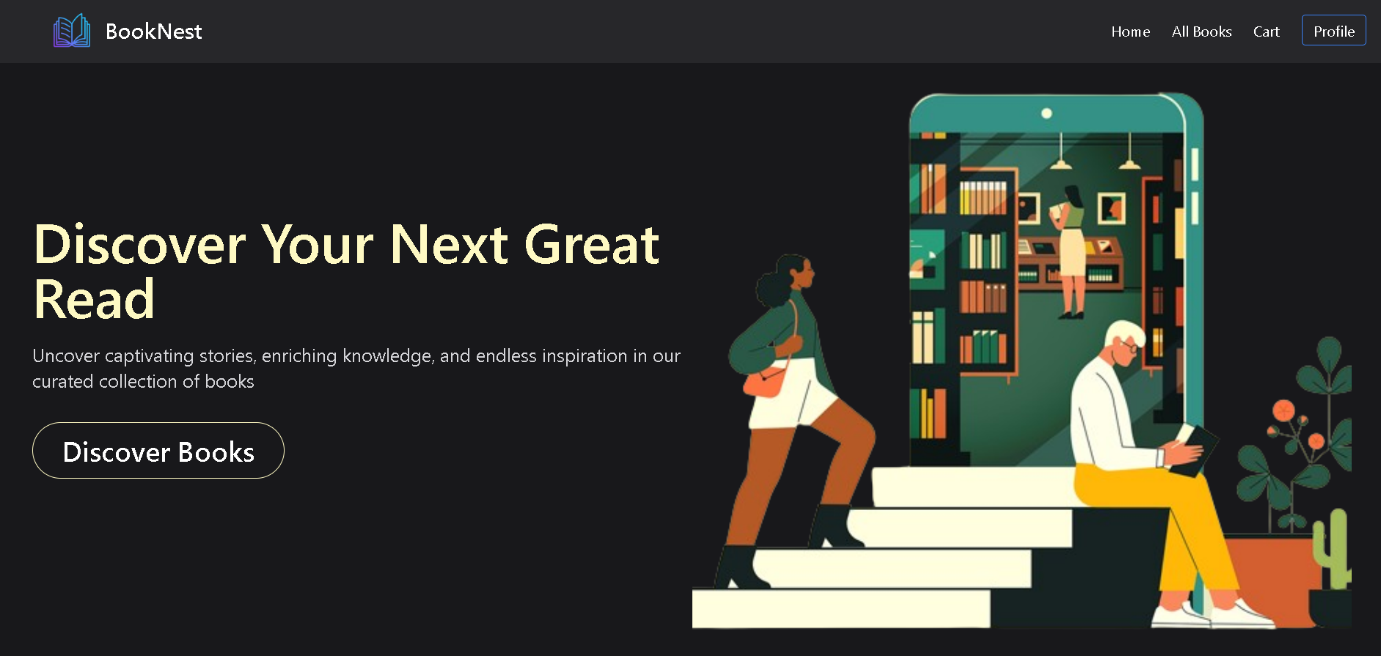
1. . **Signup Page**



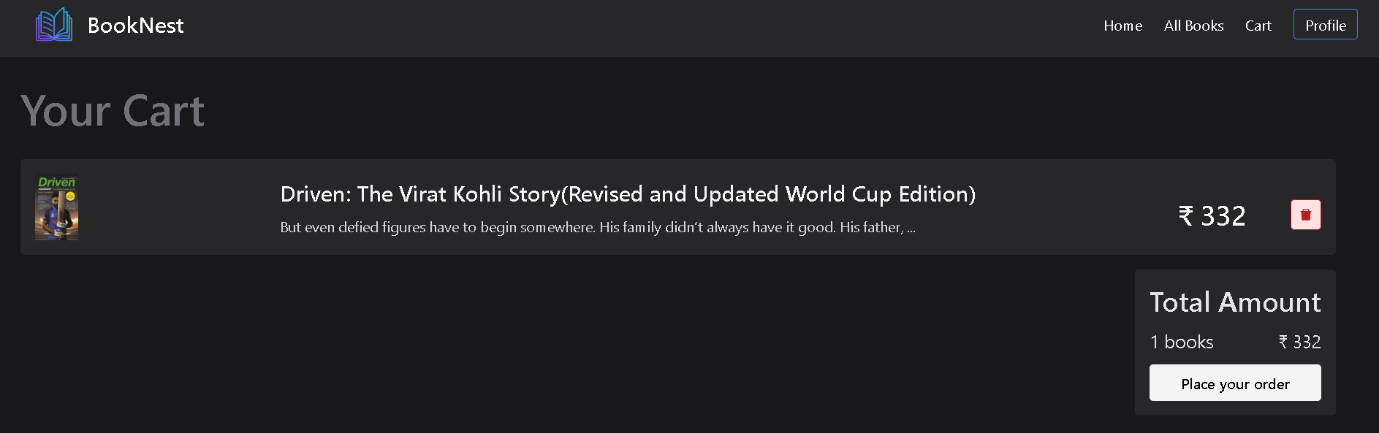
**Login page**



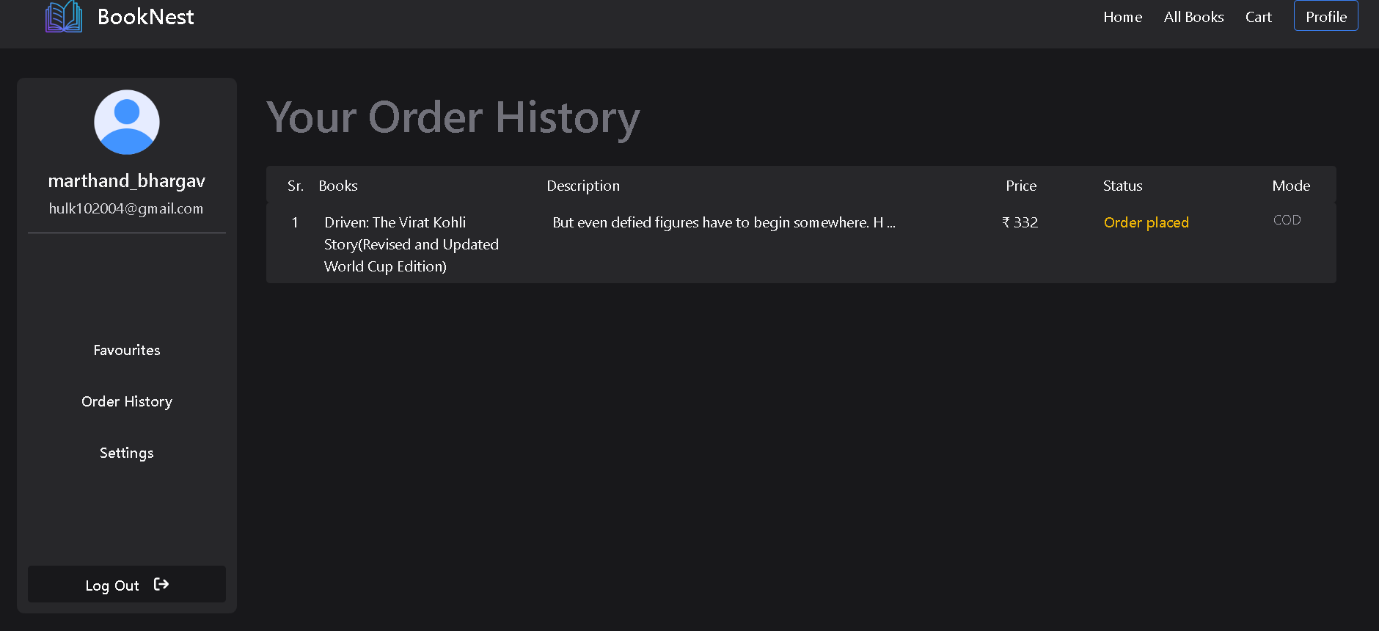
2. **Home Page**



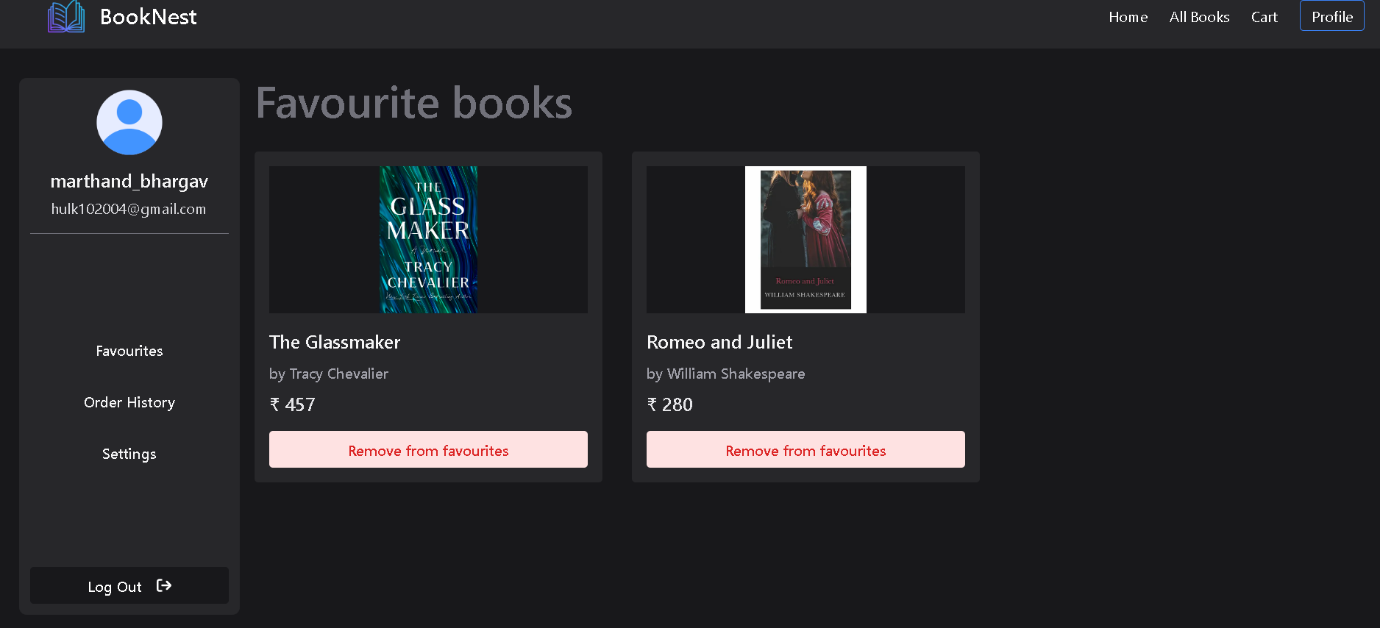
3. **Cart Page**



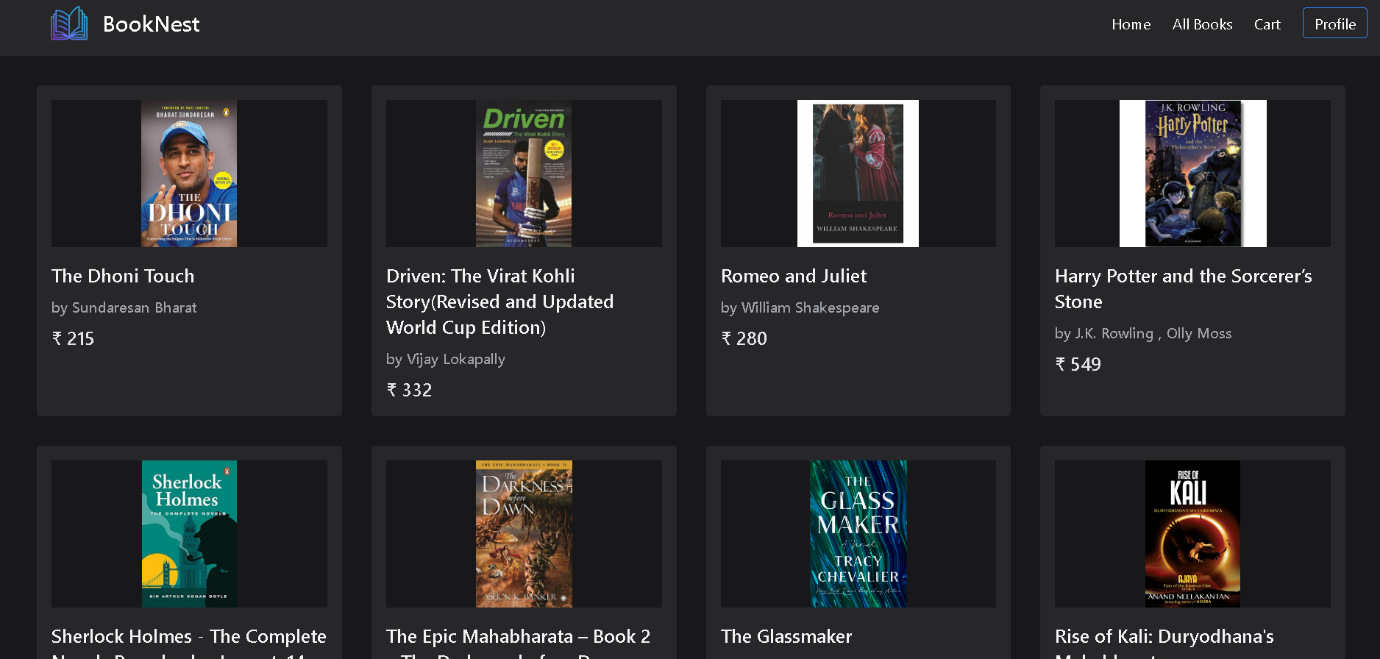
4. **Orders Page**



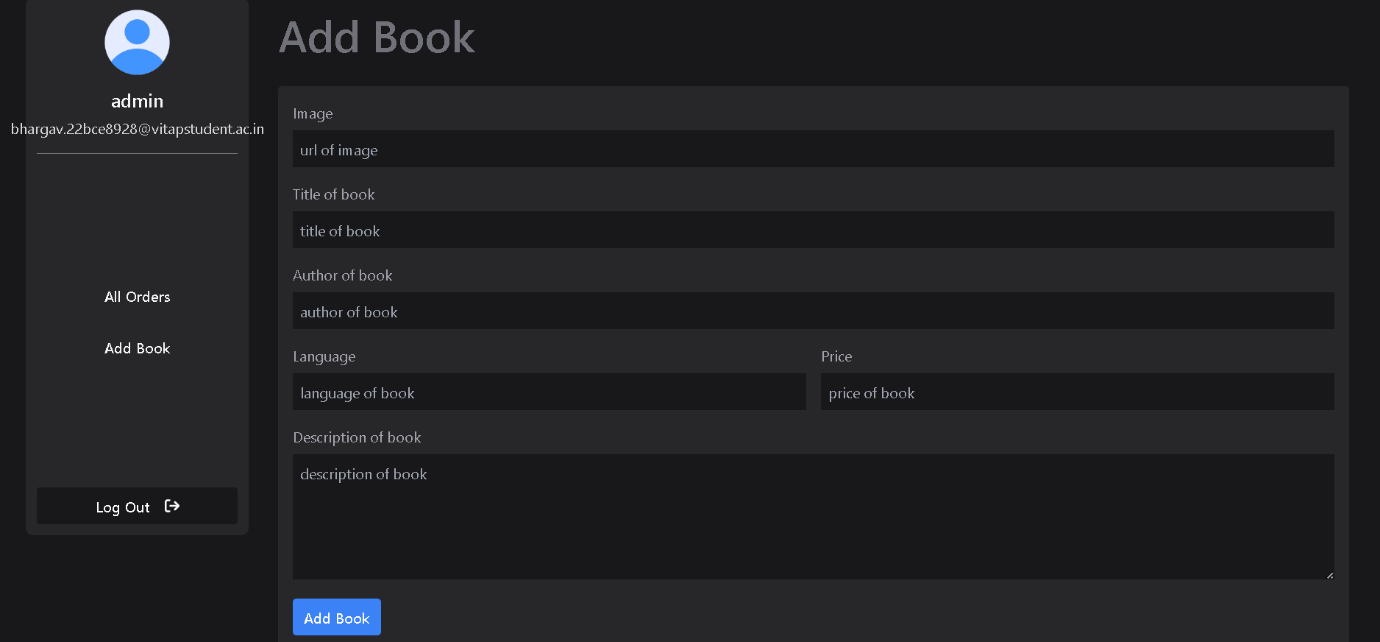
**4.Favourite books:**



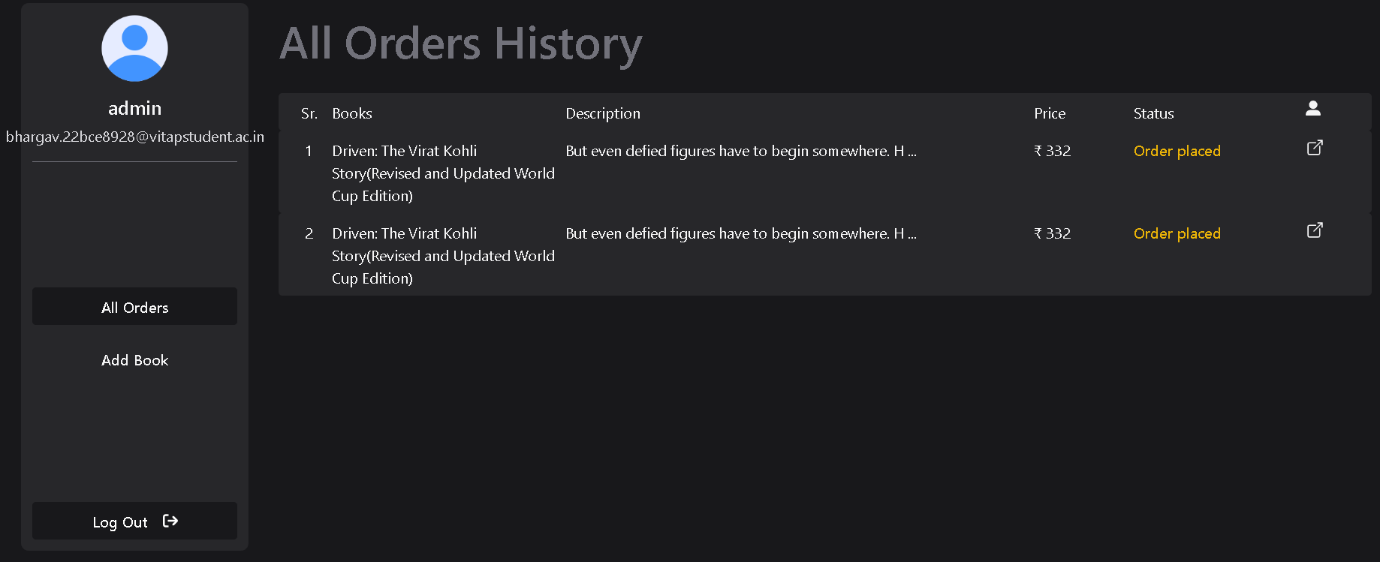
**5.All Books:**



**6.Admin Page  - for uploading books**



7. **Admin Page - for tracking orders**



**12. Known Issues**

1. Limited search functionality for books listed on the platform.
2. Payment integration feature is yet to be added

**13  . Future Enhancements**

1. Implement advanced search filters for better user experience.
2. Implement payment processing
3. Add support for user reviews and ratings for books.
4. Develop a mobile application for the BookCorner Store.

This documentation provides a comprehensive overview of the BookCorner Store project, including  its architecture, setup instructions, and key features. Future enhancements and known issues are also  outlined to guide ongoing development and improvements.