**Full Stack Development with MERN**

**Project Documentation format**

**1. Introduction**

* **Project Title:** Book Store - Website Name : BookCorner
* **Team Members:**
* Shaneel Reddy
* Vinuthna
* Mahitha
* Allwin

### 2. 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:**

* User authentication and authorization.
* Sell second-hand books with image uploads.
* Search and get books info which is fetched through the Google Play Book API.
* Manage books, orders, and users in the admin dashboard.
* Membership plan allowing unlimited book sales per month.

### 3. 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, and membership plans.

### 4. Setup Instructions

**Prerequisites:**

* Node.js
* MongoDB

**Installation:**

#### Backend Setup

1. **Create the Folder**:
   * Navigate to your desired directory and create a folder bookcorner
   * Change your directory to the backend folder: bookcorner/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 store the MONGODB\_URL and SECRET\_KEY
4. **Start the Backend Server**:
   * Use the command line to start the server: node app

#### Frontend Setup

1. **Navigate to Frontend Directory**:
   * Change your directory to the frontend folder: BookNest/frontend.
2. **Install Dependencies**:
   * Use your command line to run the command to install all necessary dependencies: npm install.
3. **Build the Frontend**:
   * Use the command line to build the frontend: npm run build.

### 5. Folder Structure

**Client:**

* /src
  + /components

Contains reusable UI components.

* + - Navbar.js - code for the navbar which is called in required pages .
    - Footer.js - contains the function for footer called when needed.
    - SideNavbar.js -function called in the admin page to display the sidenav
    - AdminOrders.js -is used to fetched the details of the order details.
    - BookComponent.js - fetches the books from google Api.
    - BookList.js - fetches the bestseller book details .
    - Footer.js - for the footer of the website.
    - HandScroller.js - for displaying best seller books in cards.
    - HeroComponent.js - for the main banner thing in the home page .
    - HorizantalCards.js - for displaying the used book info in the cards.
    - Navbar.js - for the navbar component in the website .
    - SideNavbar.js - for vertical navbar in admin page .
    - UploadBook.js - upload functionality for the admin panel.
    - UsedBookList.js - used to fetch the used books details .
  + /pages
    - Contains page components
    - HomePage.js - includes all the components from /components that should be displayed in the home page in order.
    - Signup.js - code for a form to be displayed with name , email and password so that user can signup
    - Admin.js-the page for admin panel
    - CartsPage.js-the page where user can add items to cart and place orders
    - Login.js-the user can give mail ,password and can login into the website.
    - MembershipPage.js-this page contains the details about the membership plans for selling books.
    - OrdersPage.js - this will display all the orders placed by the customer.
    - SearchPage.js - when the user searches the books it will direct them to this page with all the matched book details.
    - Sell.js-in this page the user can sell their books .

/css\_files

Contains the styling in css for all different pages (Login.css).

* + - Login.css- styling for login and signup page like bordering , font,padding, margin etc .
    - SideNavbar.css -css styling for the navbar in the admin page like width and height of the navbar .
  + App.js
    - Complete application structure and routing..

**Server:**

* /routes
  + adminorderRoute.js - Routing for get for fetching orders for admin panel management
  + bookRoute.js- Routing for get post for uploading books from admin panel
  + cartRoute.js - Routing for get and post methods to add and delete books in cart page.
  + gbooksRoute.js - Routing for fetching data from google books Api.
  + orderRoute.js - Routing for fetching the user orders.
  + usedbookRoute.js - Routing for posting and getting the second hand books .
  + userRouter.js - Routing for login and signup.
* /controllers
  + adminController.js - for get and post operations for the admin panel .
  + bookController.js - get and post operations for books into the bestseller part .
  + cartController.js - post , get and delete the books in the cart page .
  + gbooksController.js - get operation to fetch book info from Google Book Api
  + orderController.js - for get and post operations for order details
  + usedbookController.js- for get and post operations for used books details
  + userController.js- for get and post operations for login and signup informations
* /models
  + book.js - schema for information regarding books
  + cart.js - schema for cart details
  + gbooks.js- schema for google books api’s book’s info
  + order.js schema for order details
  + usedbook.js- - schema for used books
  + user.js- schema for user data
* App.js
  + Connects to the server and database

### 6. Running the Application

**Frontend:**

* cd frontend - npm start

**Backend:**

* cd backend - npm start

### 7. API Documentation

**Endpoints:**

* **GET /getBooks**

Retrieves book data for display from database.

**Response:**

**{**

title: { type: String, required: true },

author: { type: String, required: true },

imageUrl: { type: String, required: true },

category: { type: String, required: true },

description: { type: String, required: true },

price: { type: Number, required: true }

**}**

**POST /postbooks**

adds a new book in the database under bestsellers .

**Request:**

**{**

title: { type: String, required: true },

author: { type: String, required: true },

imageUrl: { type: String, required: true },

category: { type: String, required: true },

description: { type: String, required: true },

price: { type: Number, required: true }

**}**

**POST /login**

Authenticates a user.

**Request:**

{

name: { type: String, required: true },

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 },

Address:{ type: String, required: true },

city:{ type: String, required: true },

state:{ type: String, required: true },

zip: { type: String, required: true },

Phone:{ type: String, required: true },

price: { 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 },

city:{ type: String, required: true },

state:{ type: String, required: true },

zip: { type: String, required: true },

Phone:{ type: String, required: true },

price: { type: Number, required: true }

}

**8. 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.

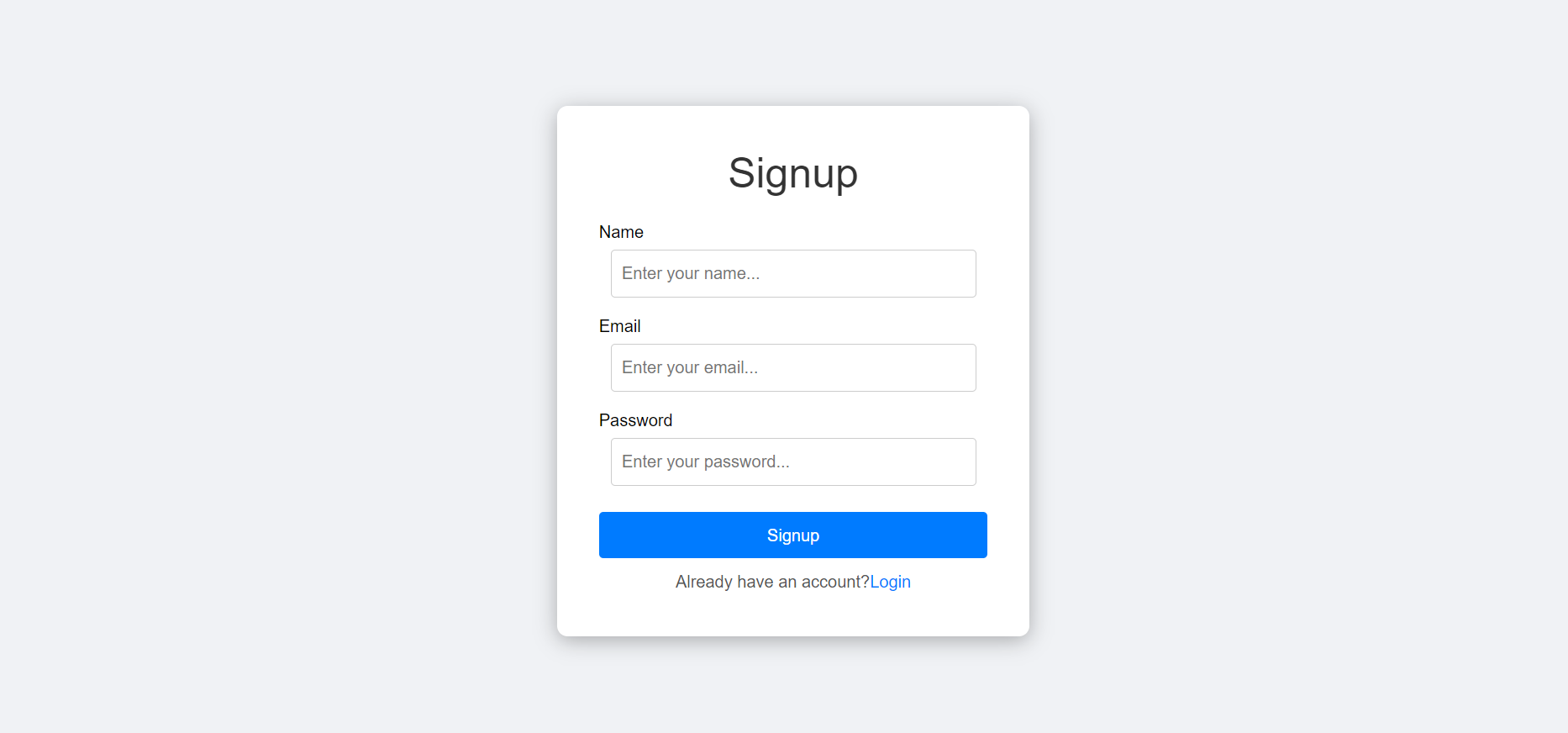
### 9. User Interface

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

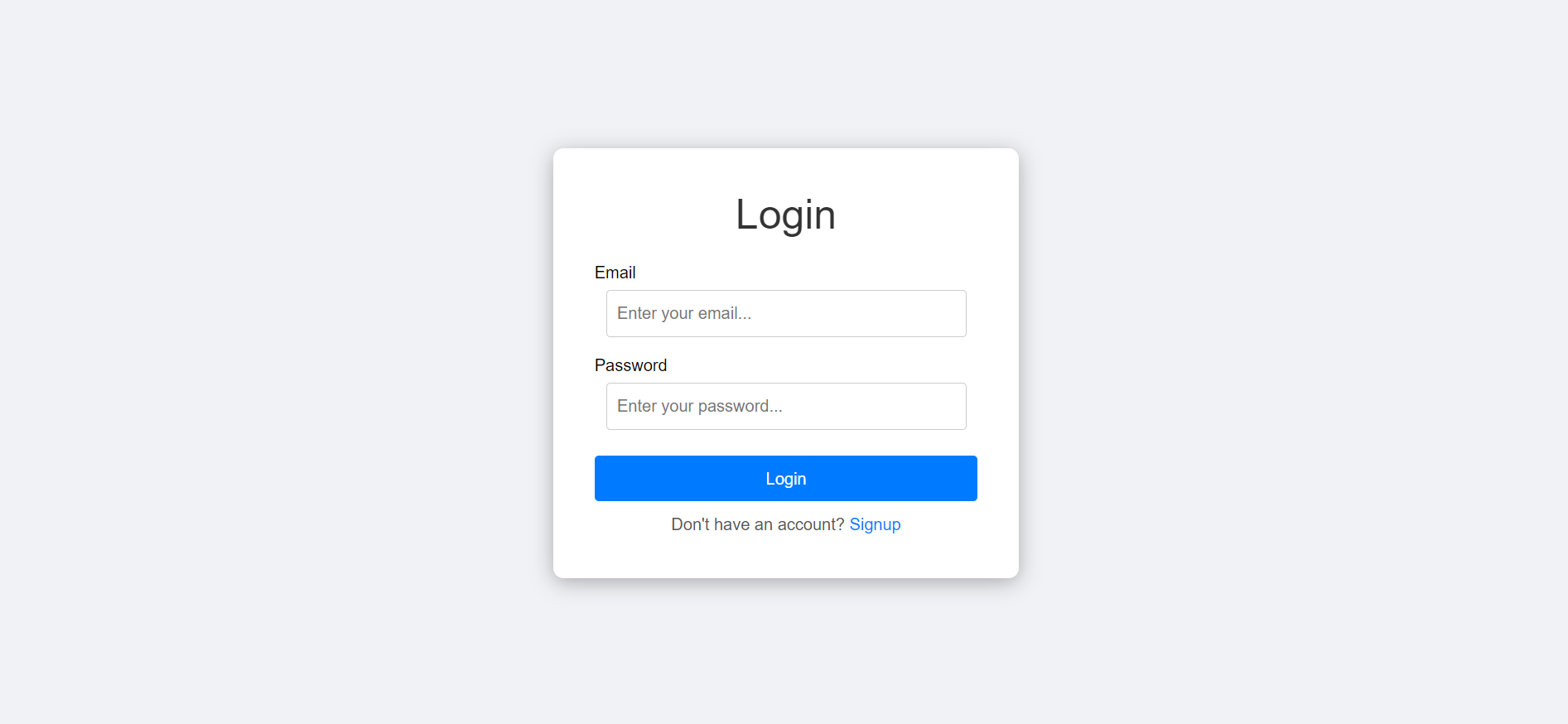
### 10. Testing

Manual test is performed and has passed all test cases.

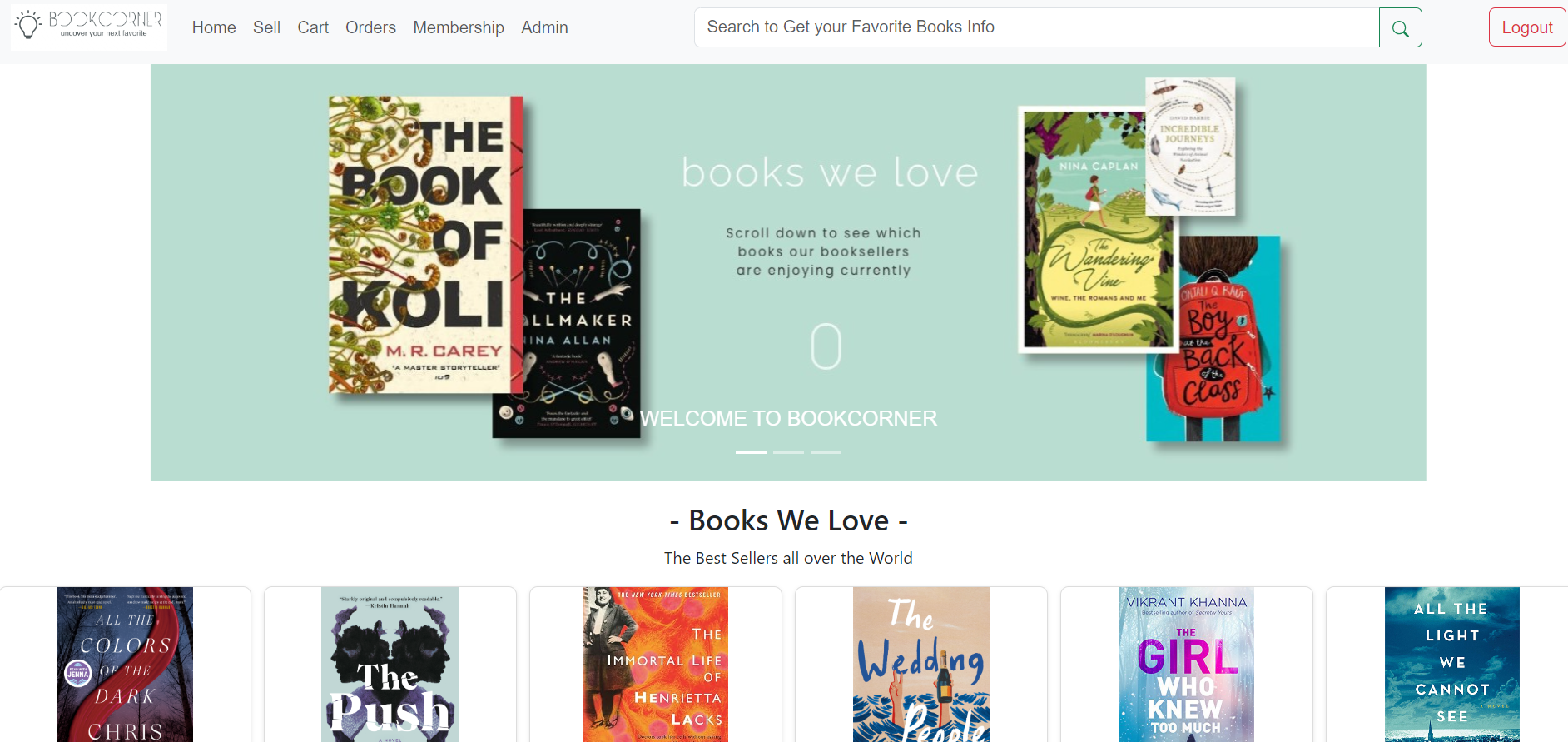
**1**. **Signup Page**

****

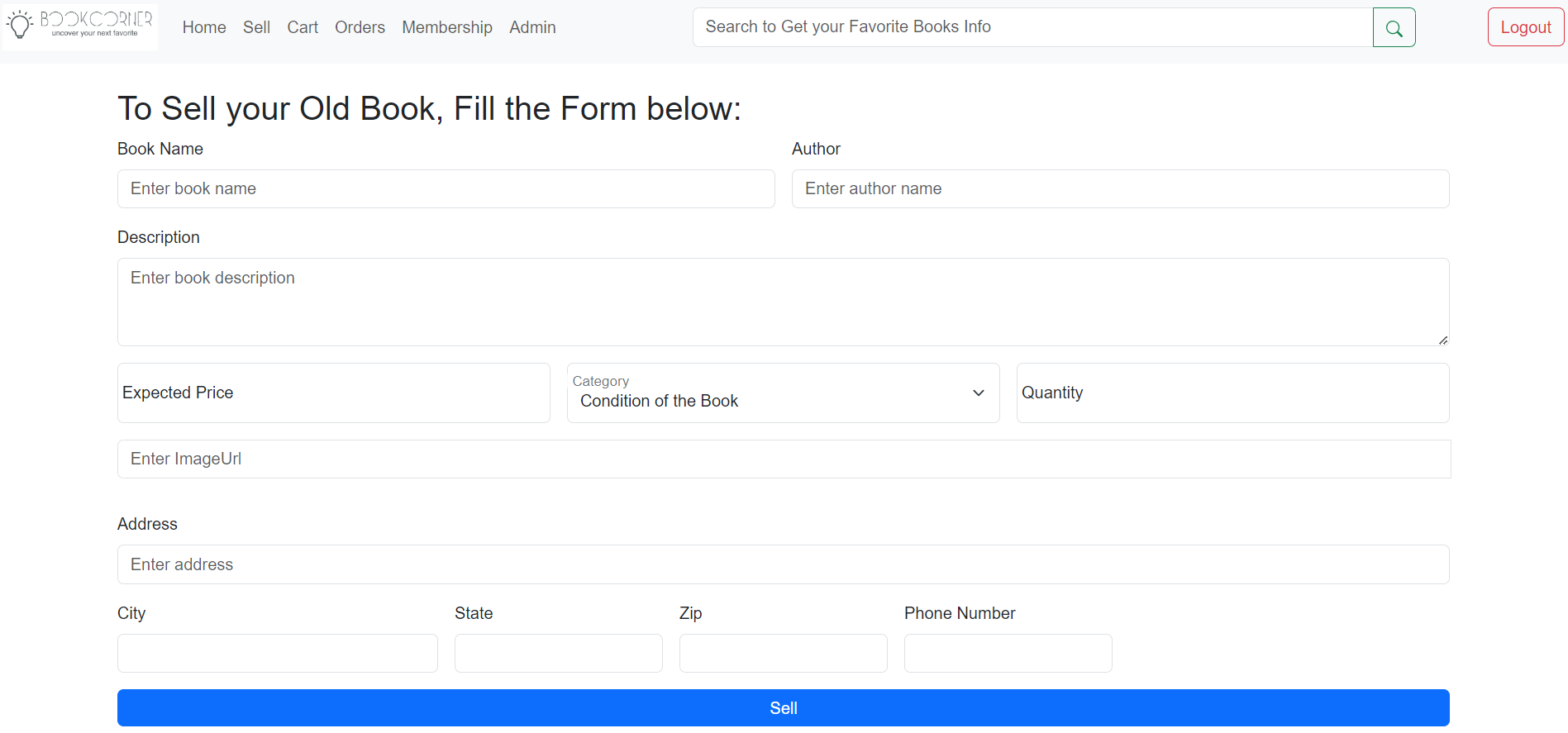
**2**. **Login page**

****

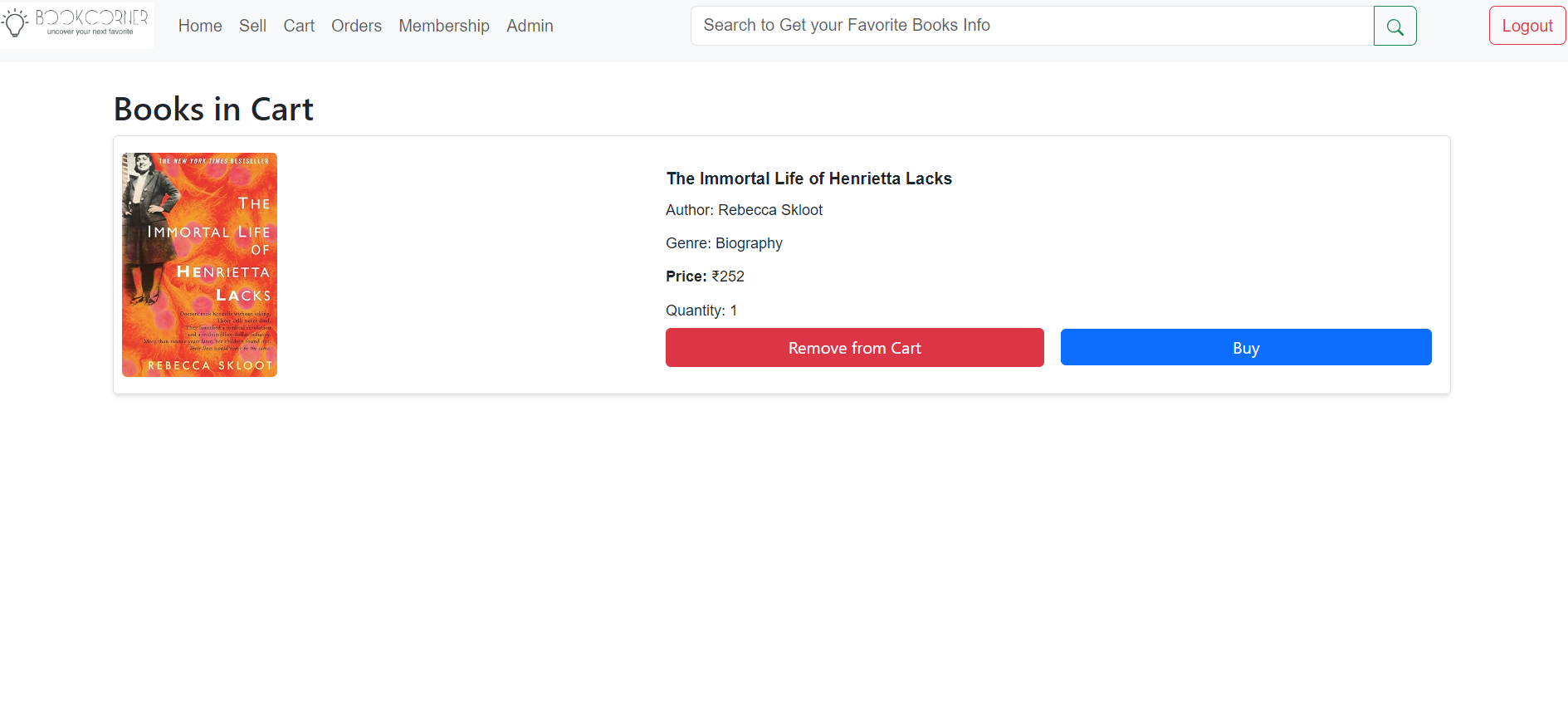
**3**. **Home Page**

****

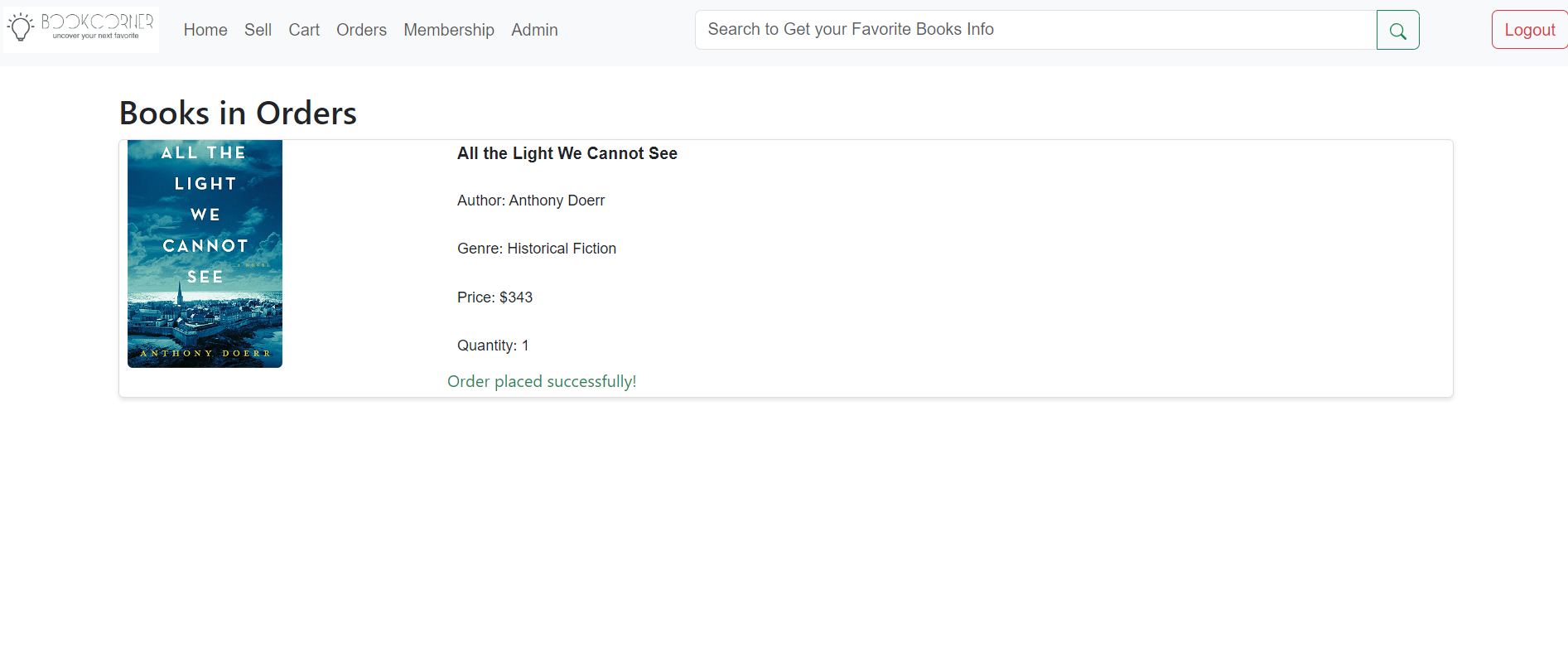
**4**. **Sell Page**

****

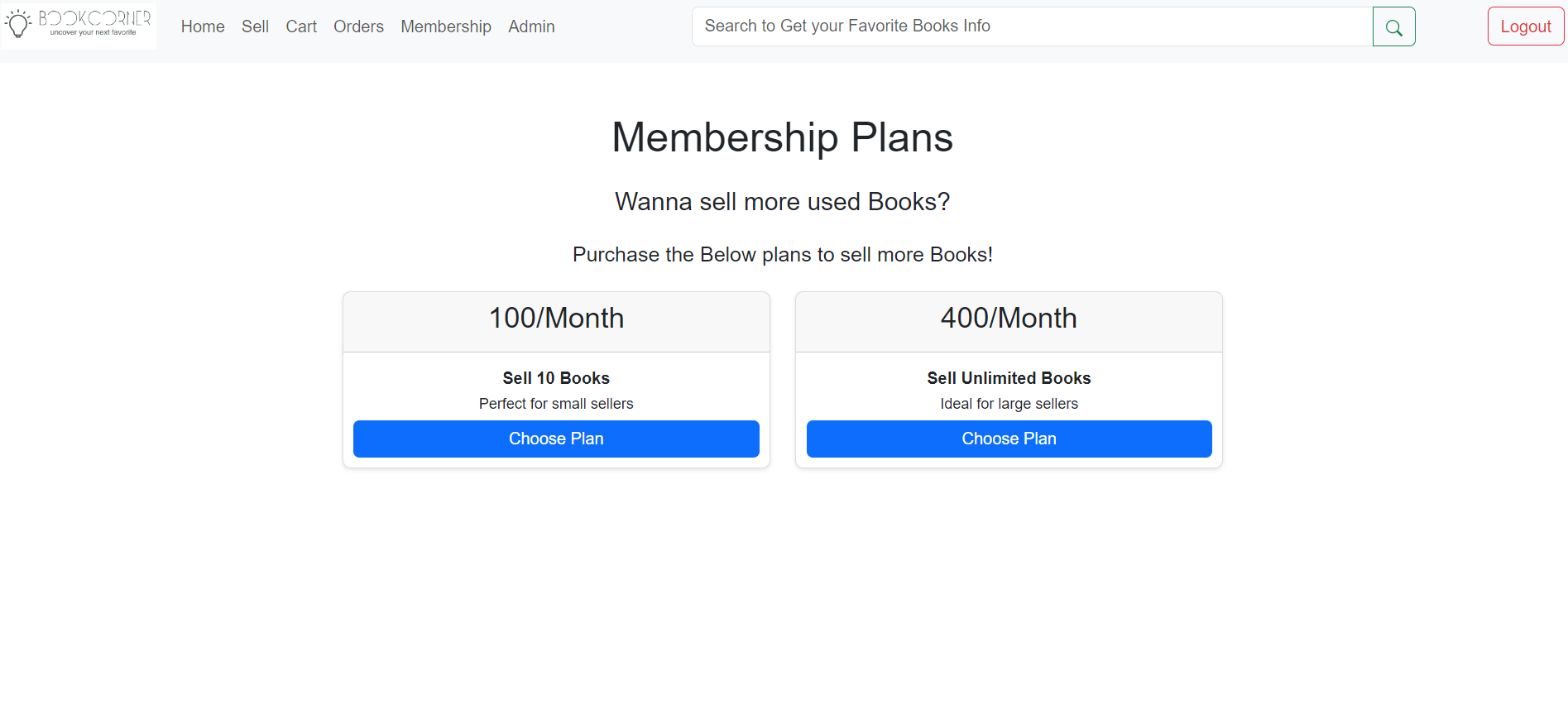
**5**. **Cart Page**

****

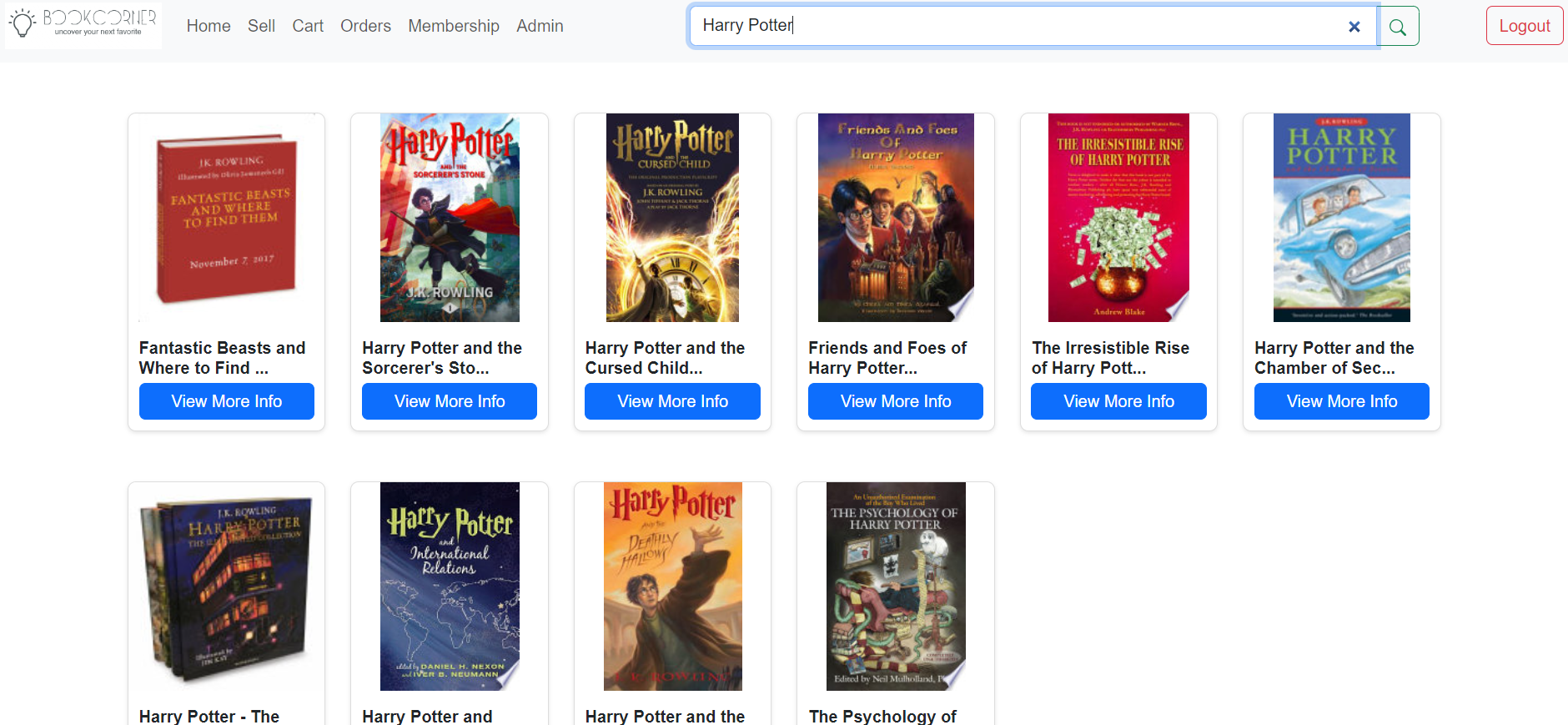
**6** . **Orders Page**

****

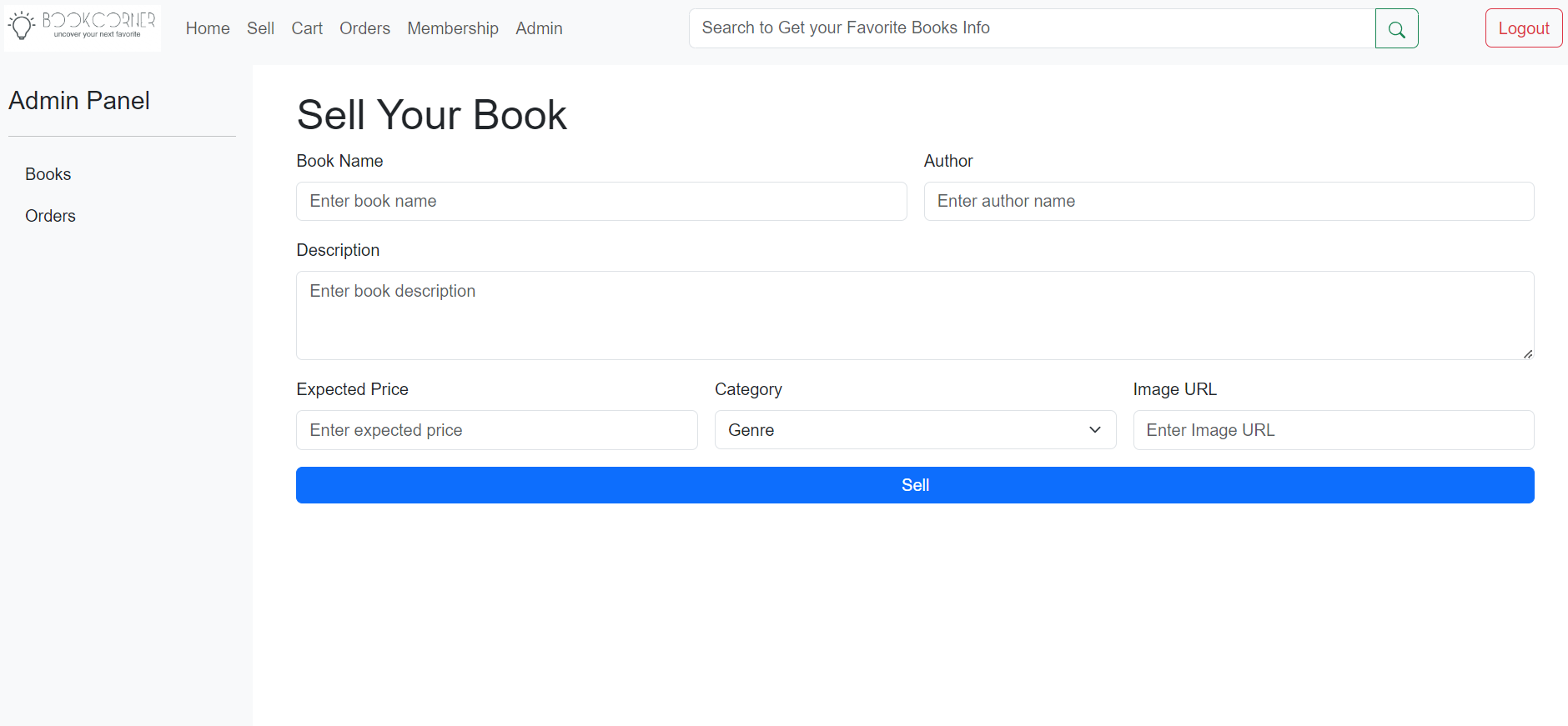
**7**. **Membership Page**

****

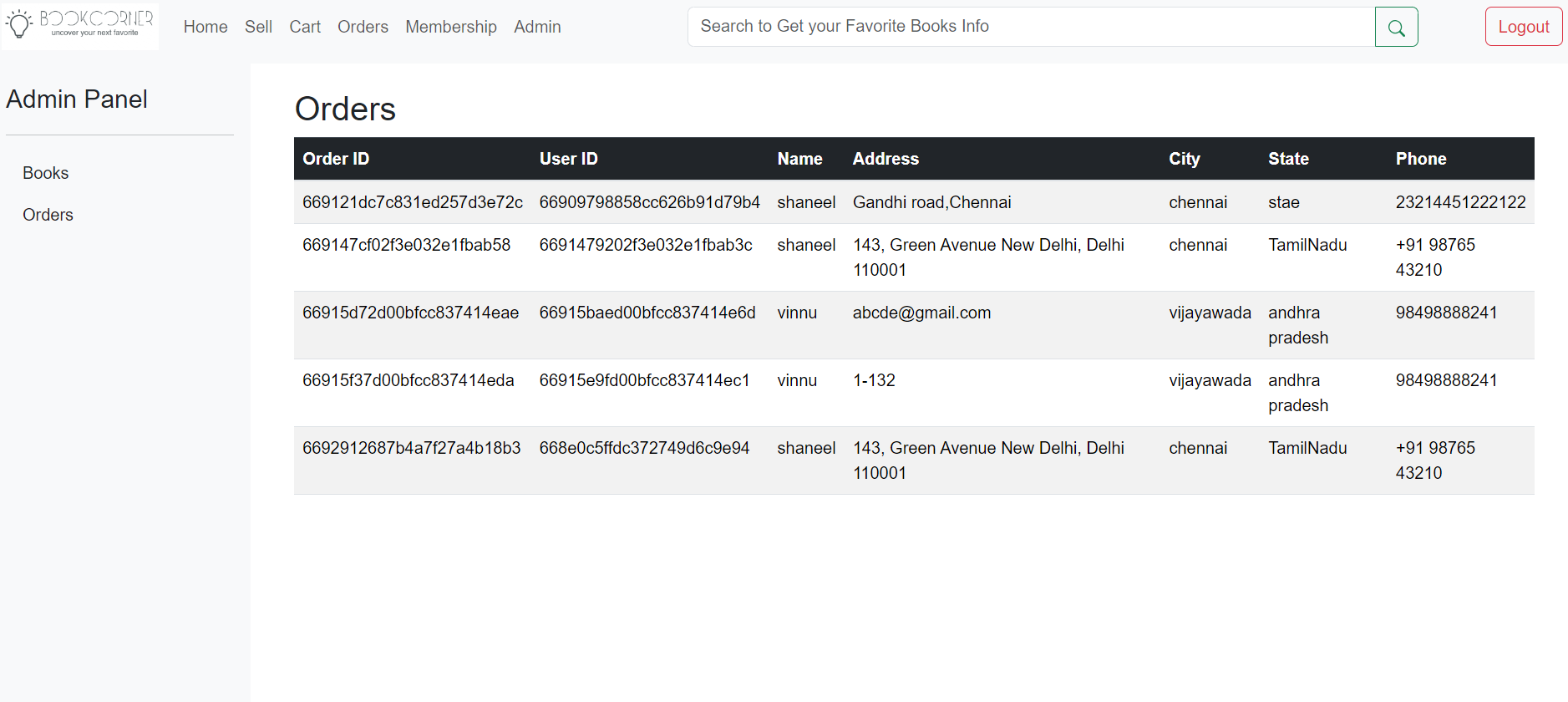
**8**. **Search Page**

****

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

****

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

****

### 12. Known Issues

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

### 13. Future Enhancements

* Implement advanced search filters for better user experience.
* Implement payment processing
* Add support for user reviews and ratings for books.
* 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.