# Full Stack Development with MERN

## Project Documentation

## I. Introduction

**Project Title:** House Rent App

**Team Members:**

**Amreen Fathima A** – Project Manager and Testing: Manages the project, leads the team, ensures timelines, and oversees testing.

**Dharshini K** – Backend Developer: Develops server-side logic, APIs, and database interactions.

**Archanaa M** – Database Administrator: Maintains and optimizes databases, ensuring integrity and performance.

**Lufiya Banu A** – Frontend Developer: Designs responsive interfaces, ensuring seamless user experiences.

## II. Project Overview

**Purpose:**

The House Rent App simplifies the process of listing, searching, and managing rental & sales properties, offering an intuitive and efficient experience for both owners and renters.

**Key Features:**

**-Property Listing & Management:** Users can easily add, edit, or remove property listings, allowing them to keep their listings up-to-date and relevant.

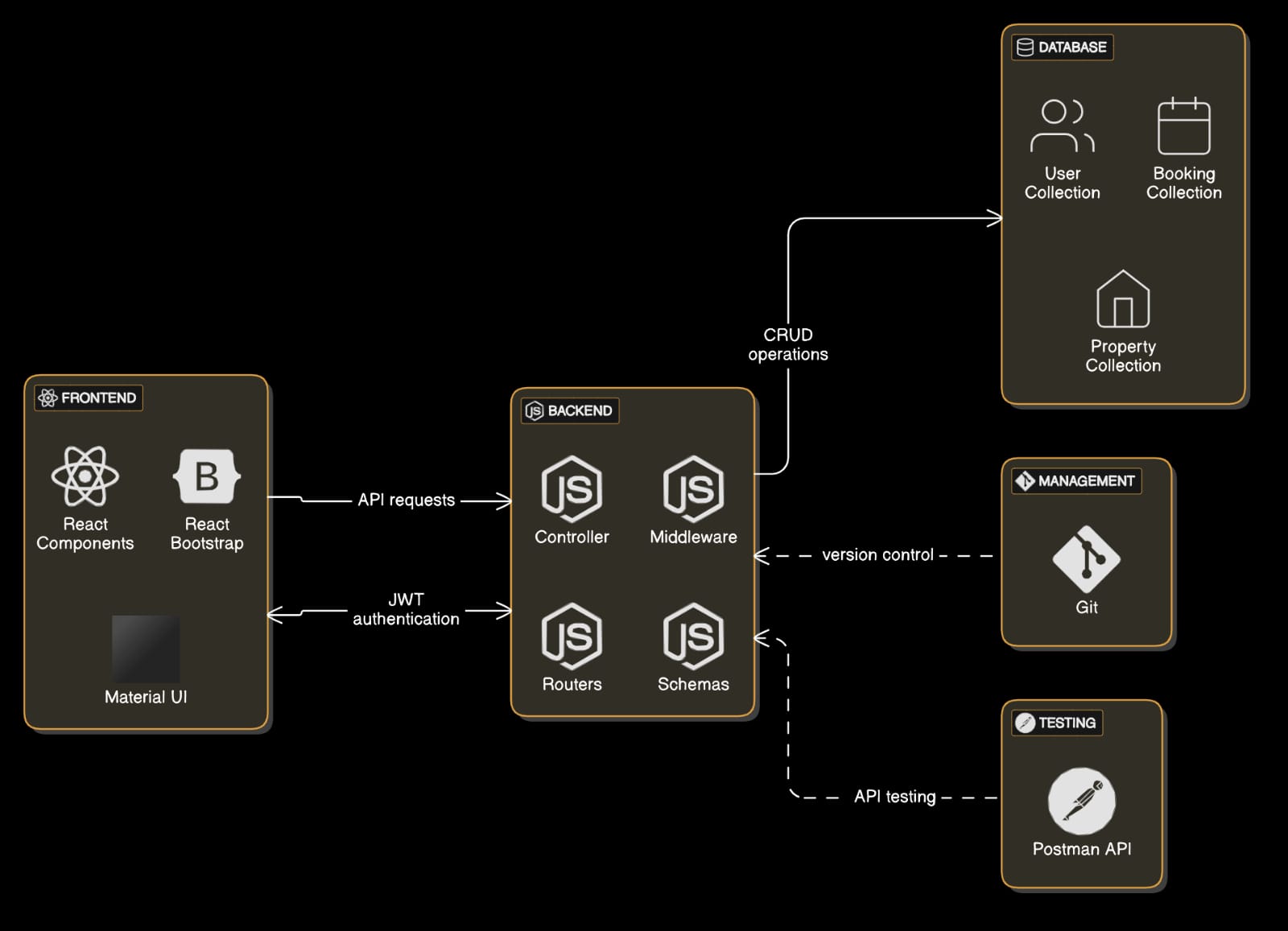
**-Search & Filter Options:** Users can refine property searches by location, property type, amenities, and more, ensuring they find the ideal match.

**-Instant Booking:** Request to book a property instantly.

**-User profiles:** Dedicated profiles for owners and renters with personalized dashboards.

**-Secure Authentication:** - Users receive a JWT upon login, stored in local storage.

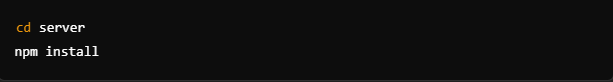
## III. Architecture



## IV. Setup Instructions

**Prerequisites:**  
- Node.js  
- MongoDB Compass  
- npm (Node Package Manager)

**Installation Steps:**  
1. Clone the repository:

  
2. Install dependencies:  
  **Frontend:**  
  
  **Backend:**  
  
3. **Set up environment variables:**  
 - Create a .env file in the server directory, adding necessary variables (e.g., MongoDB URI, API keys).

## V. Folder Structure

**1. Frontend:**  
 - /src: Main directory with React components, pages, and services.  
 - /src/modules: Feature-specific modules including:  
 - Admin: Admin dashboard components and pages.  
 - Common: Shared components for reuse.  
 - User: Components for user functionalities:  
 - Owner: Property owner-related components.  
 - Renter: Renter-related components.

**2. Backend:**  
 - /config: Configuration files.  
 - /controllers: Request handling logic.  
 - /middlewares: Authentication and error-handling middleware.  
 - /routes: API route definitions.  
 - /schemas: Mongoose schemas.  
 - /uploads: Temporary file storage.  
 - index.js: Initializes the Express app.

## VI. Running the Application

**Frontend:**

cd frontend  
npm start

**Backend:**

**Backend:**

cd backend  
npm start

**VII. API Documentation**

**Base URL:** All endpoints are prefixed with /api.

1. **User Routes** **(/api/user)**:
   1. **Register User:**

* Endpoint: /register
* Method: POST
* Parameters:
* Body:

{"name": "John Doe", "email": "johndoe@example.com", "password": "password123", "type": "Owner"}

* Response:
* Success:

{"message": "Registered Successfully","success": true}

* Error:

{"message": "User already exists","success": false}

* 1. **Login User:**
* Endpoint: /login
* Method: POST
* Parameters:
* Body:

{"email": "johndoe@example.com", "password": "password123"}

* Response:
* Success:

{"message": "Login successfully", "success": true, "token":"<JWT\_TOKEN>", "user": {"\_id": "64fdbb4f4321...", "email": "johndoe@example.com", "type": "Owner",}}

* Error:

{"message": "Invalid email or password", "success": false}

* 1. **Forgot Password:**
* Endpoint: /forgotpassword
* Method: POST
* Parameters:
* Body:

{"email": "johndoe@example.com","password": "newpassword123"}

* Response:
* Success:

{"message": "Password changed successfully","success": true}

* Error:

{"message": "User not found","success": false}

* 1. **Get All Properties:**
* Endpoint: /getAllProperties
* Method: GET
* Headers:
* Authentication: Bearer <JWT\_Token>
* Response:
* Success:

{"success": true,"data": [{"\_id": "64fe1c1e...","propertyType": "Rent", "propertyAddress": "123 Main St","propertyAmt": 50000,...},...]}

* Error:

{"message": "auth error", "success": false}

* 1. **Booking Handle Controller:**
* Endpoint: / bookinghandle/:propertyid
* Method: POST
* Parameters:
* Path: propertyid
* Body:

{"userDetails": {"fullName": "John Doe","phone": "1234567890"}, "status": "Pending", "userId": "64fdbb4f4321...", "ownerId": "64fdbb4f4322..."}

* Response:
* Success:

{"message": "Booking status updated","success": true}

* Error:

{"message": "Error handling booking","success": false}

* 1. **Get All Bookings Controller:**
* Endpoint: / getallbookings
* Method: GET
* Parameters:
* Body:

{“userId": "64fdbb4f4321..."}

* Response:
* Success:

{"success": true,"data": [{"propertyId": "64fe1c1e...", "userName":"John Doe", "phone": "1234567890", "bookingStatus": "Pending",...},...]}

* Error:

{"message": "Internal server error", "success": false}

* 1. **Auth Controller:**
* Endpoint: / getuserdata
* Method: POST
* Parameters:
* Body:

{"userId": "64fdbb4f4321..."}

* Response:
* Success:

{"success": true,"data": {"\_id": "64fdbb4f4321...","email": "johndoe@example.com","type": "Owner",...}}

* Error:

{"message": "auth error", "success": false}

1. **Owner Routes (/api/owner):**
   1. **Add Property Controller:**

* Endpoint: /postproperty
* Method: POST
* Parameters:
* Body:

{"userId": "64fdbb4f4321...","title": "Luxury Apartment","location": "Downtown City","price": 50000,"description": "A luxurious apartment with modern amenities","propertyType": "Apartment","otherFields": "..."}

* Response:
* Success:

{"success": true,"message": "New Property has been stored"}

* Error:

{"success": false,"message": "User not found"}

* 1. **Get All Owner Properties Controller:**
* Endpoint: /user/getallproperties
* Method: GET
* Parameters:
* Body:

{"userId": "64fdbb4f4321..."}

* Response:
* Success:

{“success": true,"message": "Owner's properties","data": [{"\_id":"64fe1c1e...","title": "Luxury Apartment","location": "Downtown City","price": 50000,...},...]}

* Error:

{"success": false,"message": "Internal server error"}

* 1. **Get All Bookings Controller:**
* Endpoint: /getallbookings
* Method: POST
* Parameters:
* Body:

{"userId": "64fdbb4f4321..."}

* Response:
* Success:

{"success": true, "message": "All bookings", "data": [{"\_id": "64fdbb4f...","propertyId": "64fe1c1e...","userName": "John Doe","phone": "1234567890","bookingStatus": "Pending",...},...]}

* Error:

{"success": false,"message": "Internal server error"}

* 1. **Handle All Booking Status Controller:**
* Endpoint: /handlebookingsstatus
* Method: POST
* Parameters:
* Body:

{"bookingId": "64fde2b3...","propertyId": "64fe1c1e...","status": "booked"}

* Response:
* Success:

{"success": true,"message": "Changed the status of property to booked"}

* Error:

{"success": false,"message": "Booking or Property not found"}

* 1. **Delete Property Controller:**
* Endpoint: /deleteproperty/:propertyid
* Method: DELETE
* Parameters:
* Path: propertyid
* Response:
* Success:

{"success": true,"message": "The property is deleted"}

* Error:

{"success": false,"message": "Property not found"}

* 1. **Update Property Controller:**
* Endpoint: /updateproperty/:propertyid
* Method: PUT
* Parameters:
* Path: propertyid
* Body:

{"userId": "64fdbb4f4321...","title": "Updated Title","location": "Updated Location","price": 45000, "description": "Updated description",...}

* Response:
* Success:

{"success": true,"message": "Property updated successfully."}

* Error:

{"success": false,"message": "Property not found"}

1. **Admin Routes (/api/admin):**
   1. **Get All Users Controller:**

* Endpoint: /getallusers
* Method: GET
* Response:
* Success:

{"success": true,"message": "All users","data": [{"\_id": "64fdbb4f4321...","name": "John Doe","email": "john@example.com",...},...]}

* Error:

{"success": false,"message": "No users found or failed to retrieve users"}

* 1. **Handle Status Controller:**
* Endpoint: /handlestatus
* Method: POST
* Parameters:
* Body:

{"userid": "64fdbb4f4321...","status": "granted"}

* Response:
* Success:

{"success": true,"message": "User has been granted"}

* Error:

{"success": false,"message": "User not found or failed to update user status"}

* 1. **Get All Properties Controller:**
* Endpoint: /getallproperties
* Method: GET
* Response:
* Success:

{"success": true,"message": "All properties","data": [{"\_id": "64fe1c1e...","title": "Luxury Apartment","location": "Downtown City",...},...]}

* Error:

{"success": false,"message": "No properties found or failed to retrieve properties”}

* 1. **Get All Bookings Controller:**
* Endpoint: /getallbookings
* Method: GET
* Response:
* Success:

{"success": true,"message": "All bookings","data": [{"\_id": "64fde2b3...","propertyId": "64fe1c1e...","userName": "John Doe","phone": "1234567890","bookingStatus": "Pending",...},...]}

* Error:

{"success": false,"message": "Failed to retrieve bookings"}

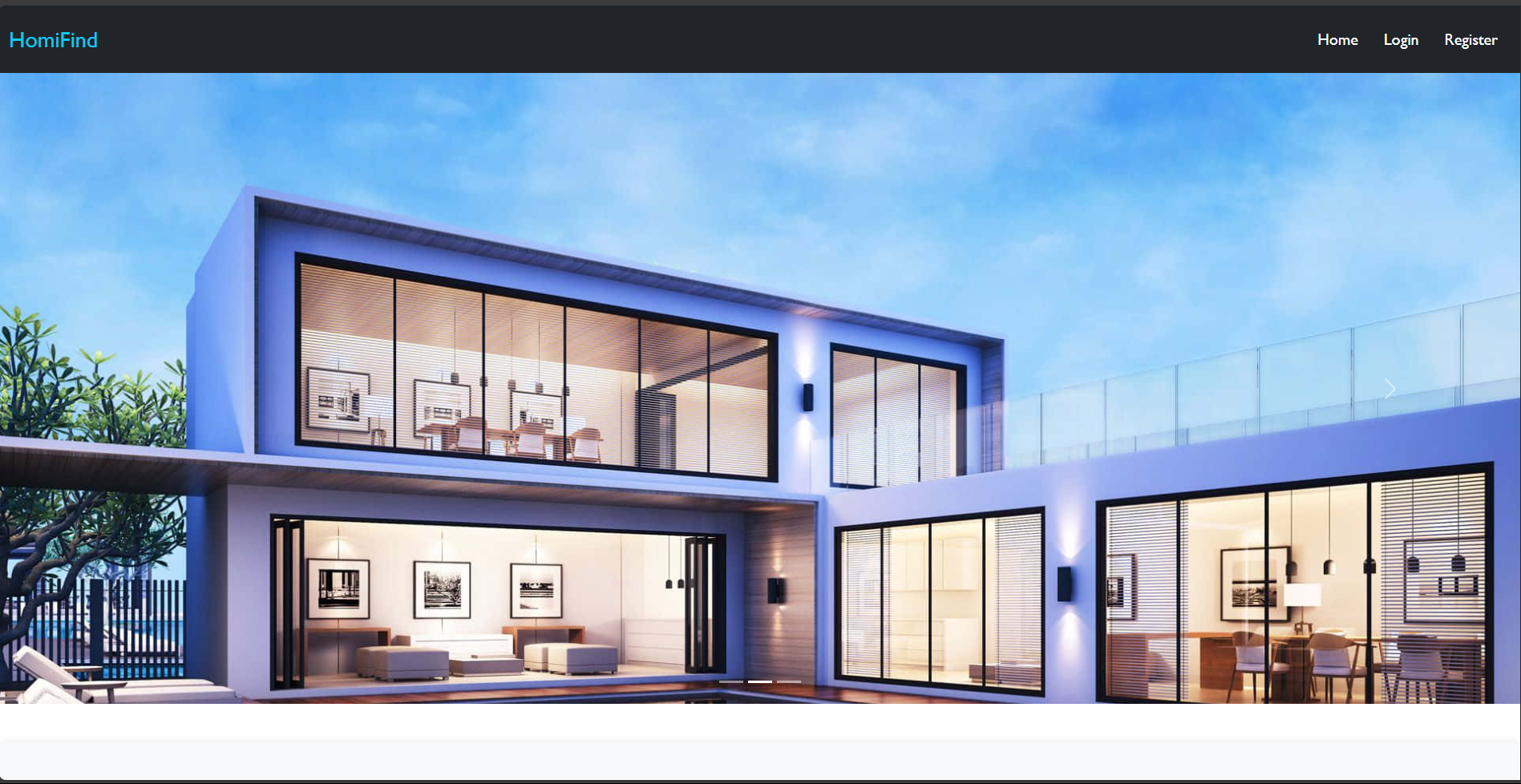
## VIII. Authentication

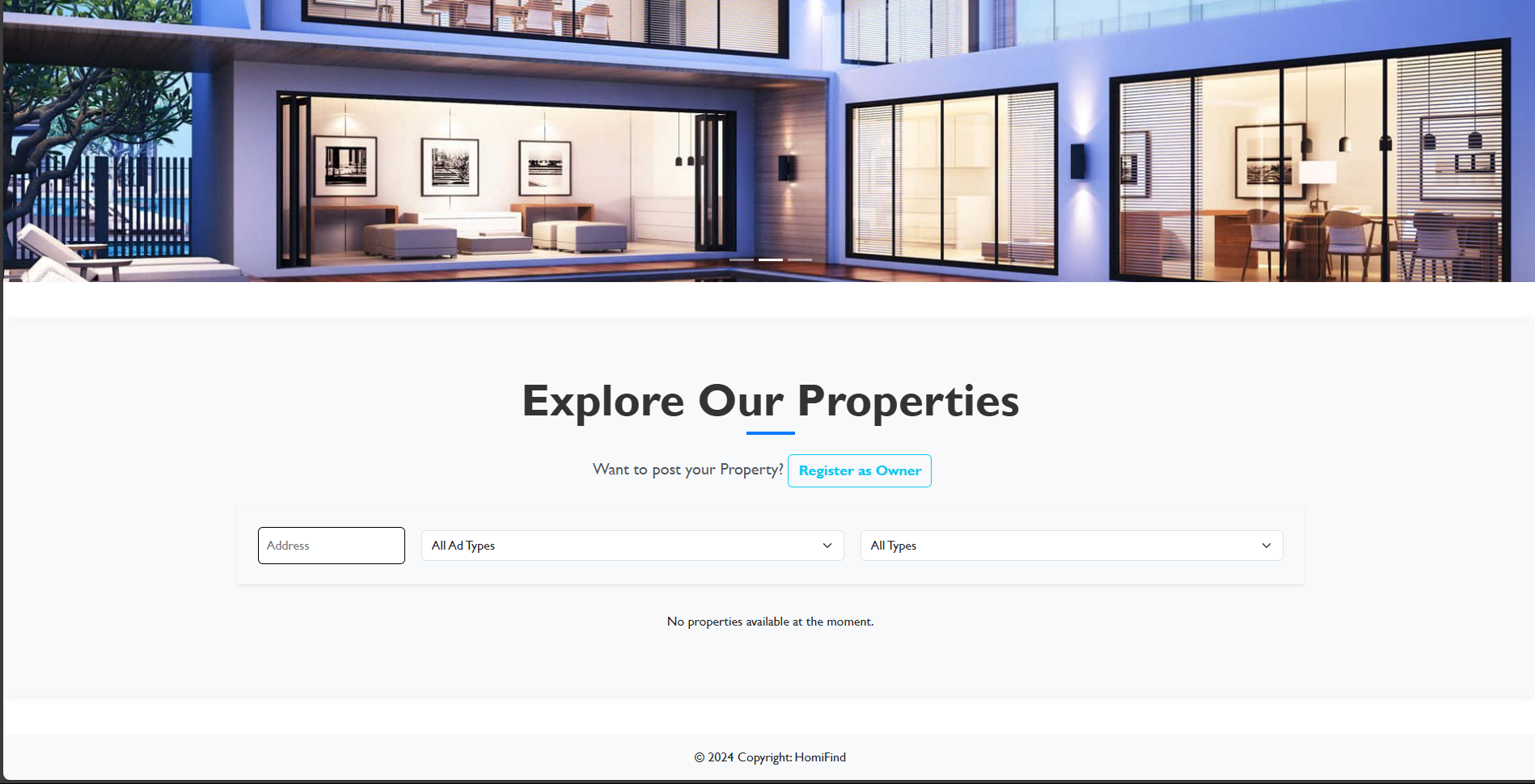
**Token-Based Authentication:**

* User receive a JSON Web Token (JWT) upon successful login, stored in local storage.
* Protected routes require a valid token, verified by a middleware function on the backend.

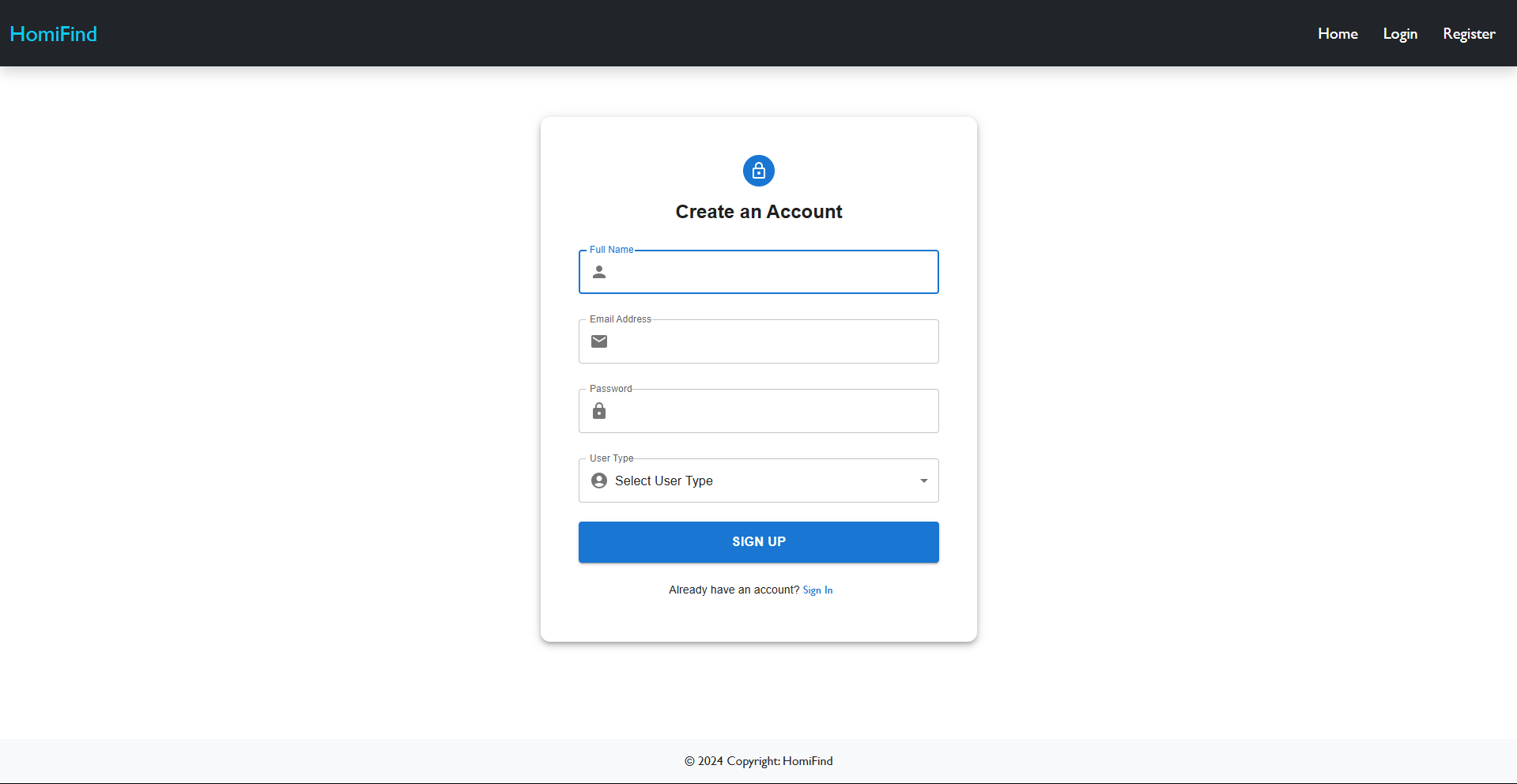
## IX. User Interface

* **HomePage**

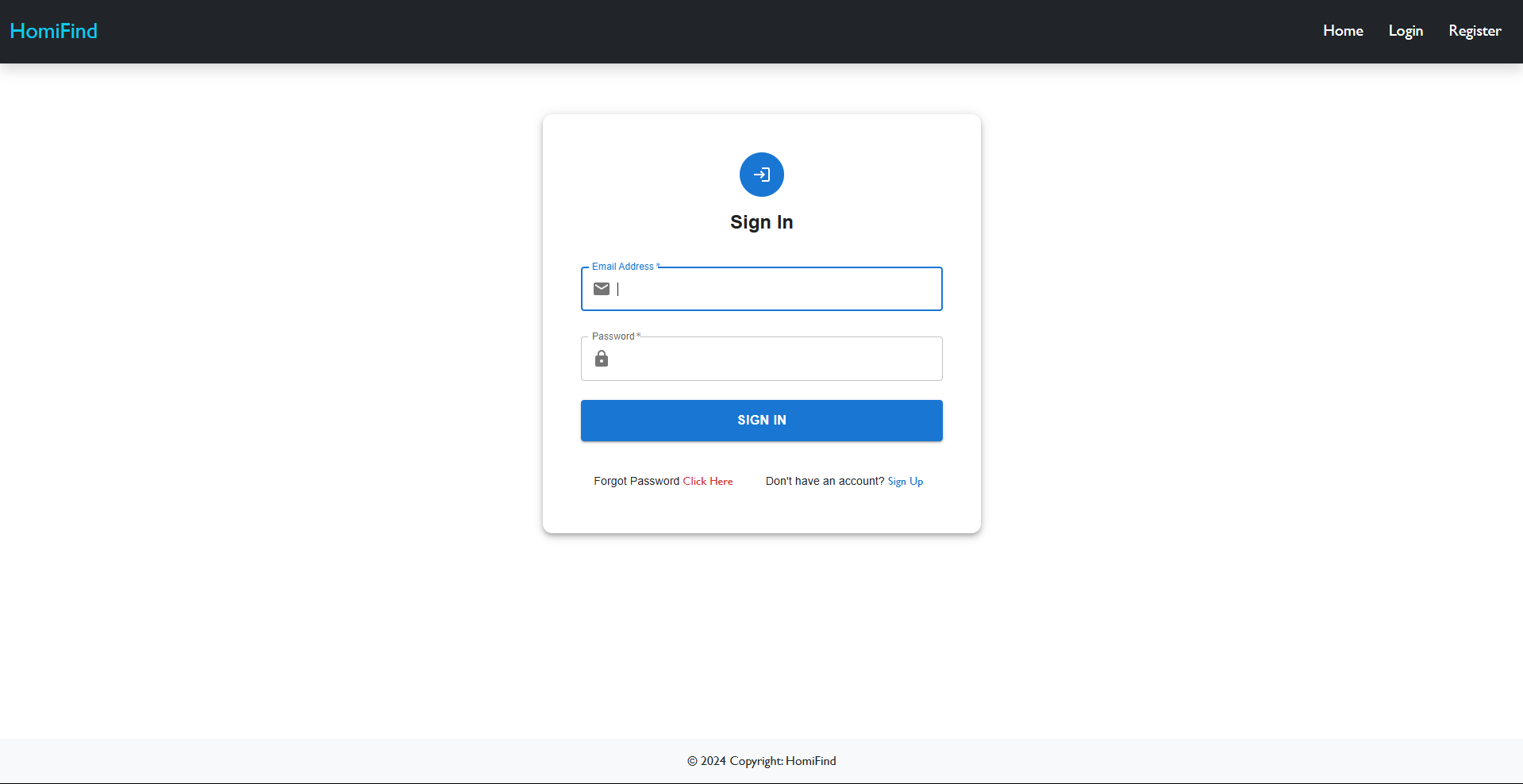




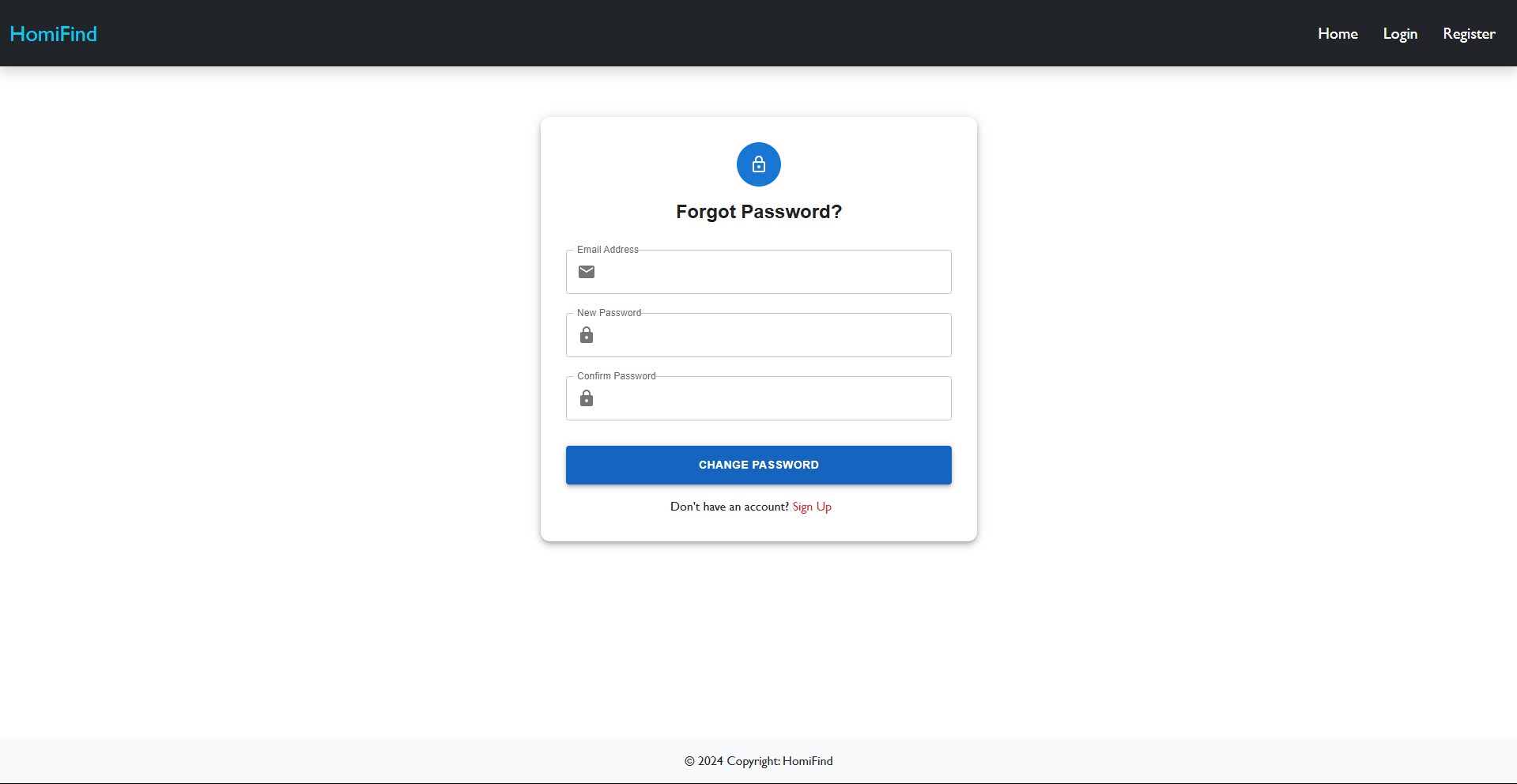
* **RegisterPage**



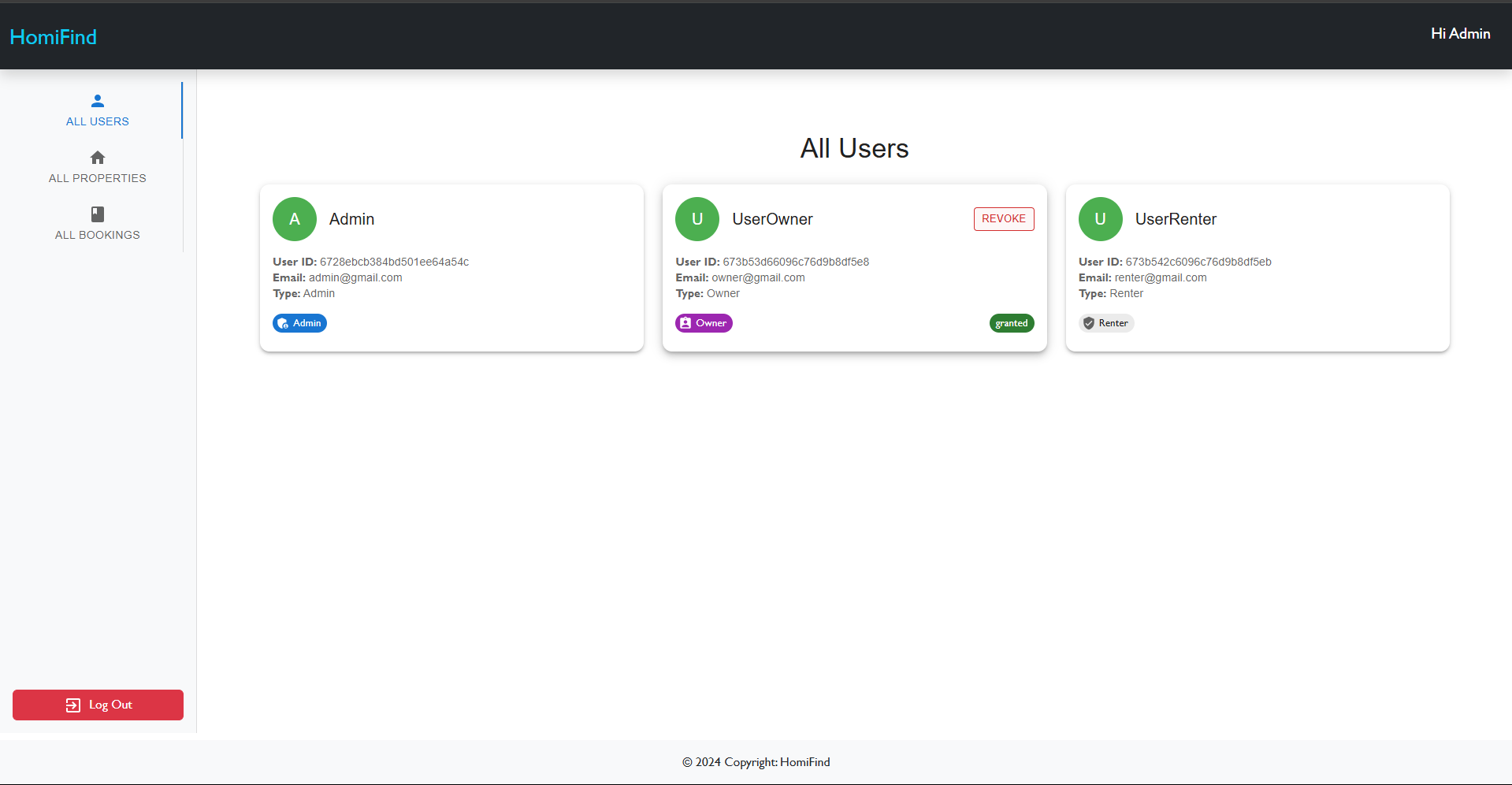
* **LoginPage**



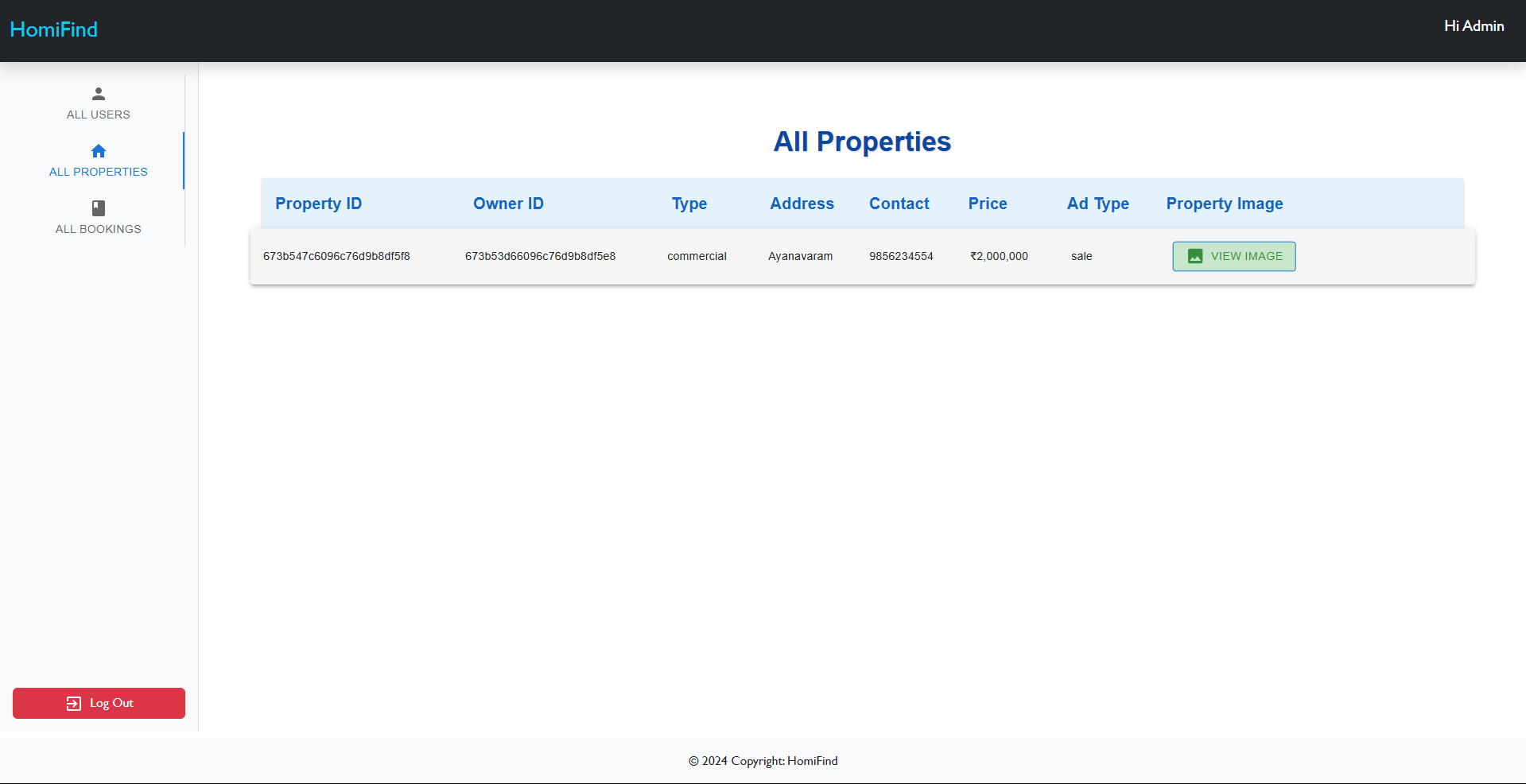
* **ForgotPasswordPage**



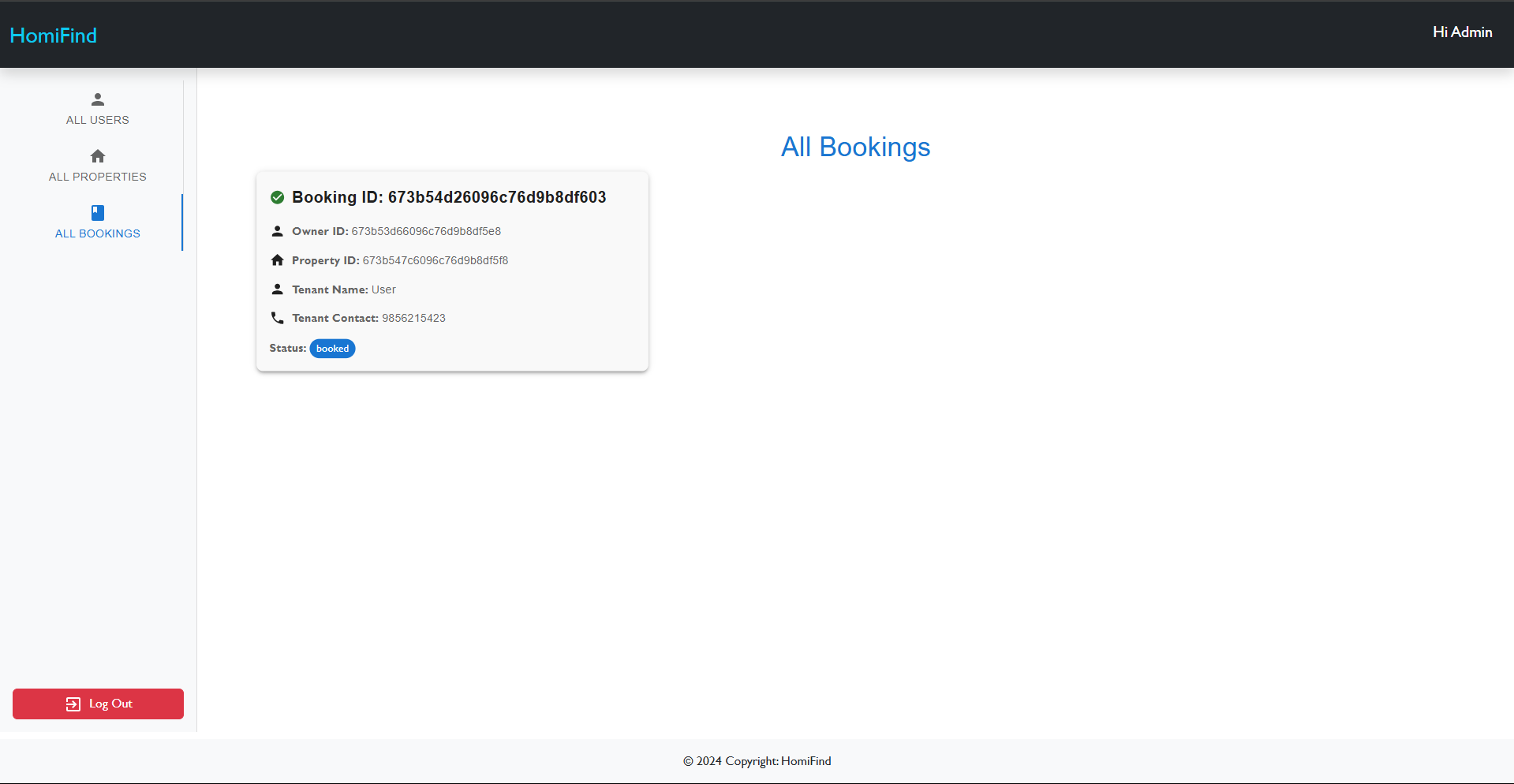
* **Admin UserPage**



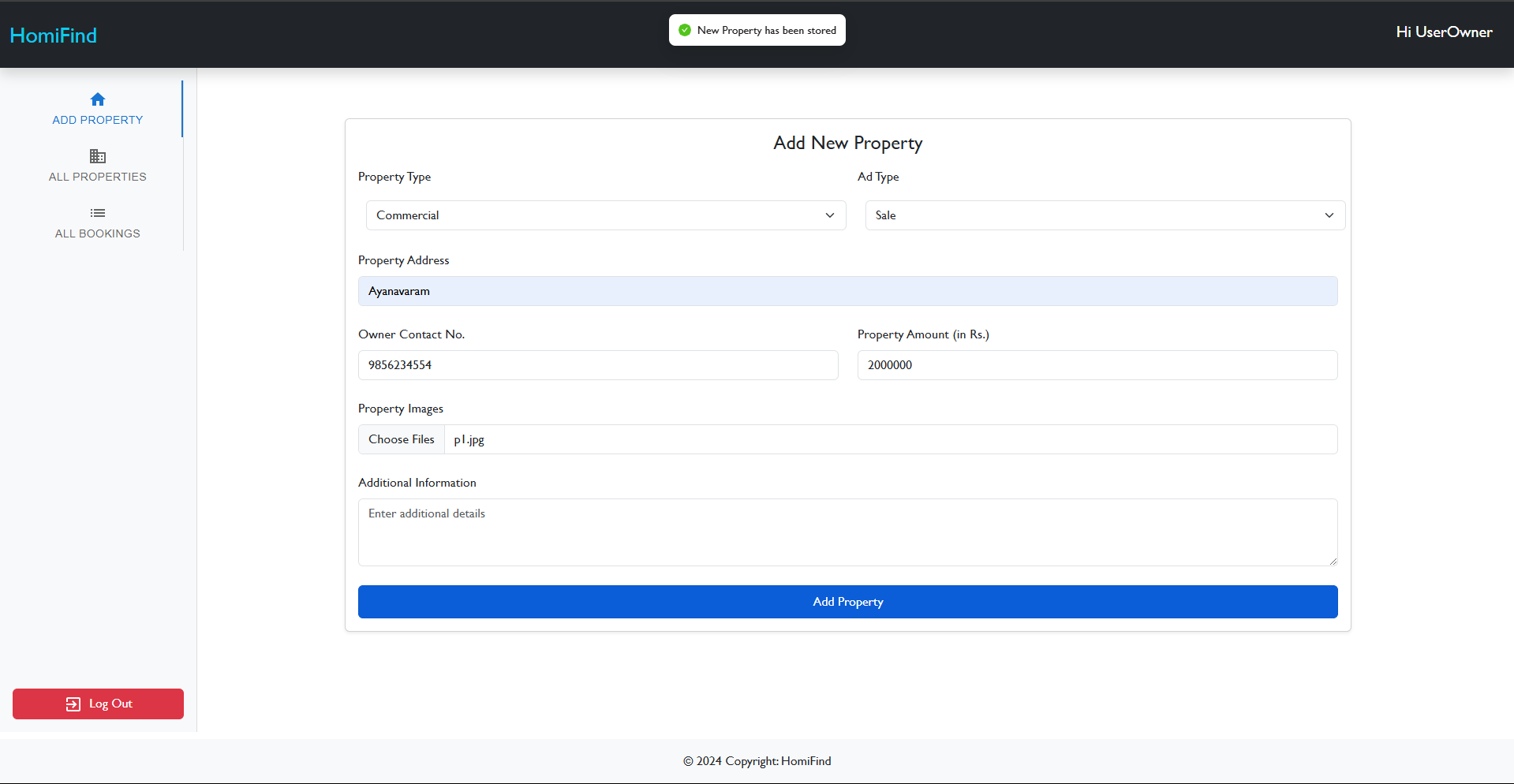
* **Admin PropertyPage**



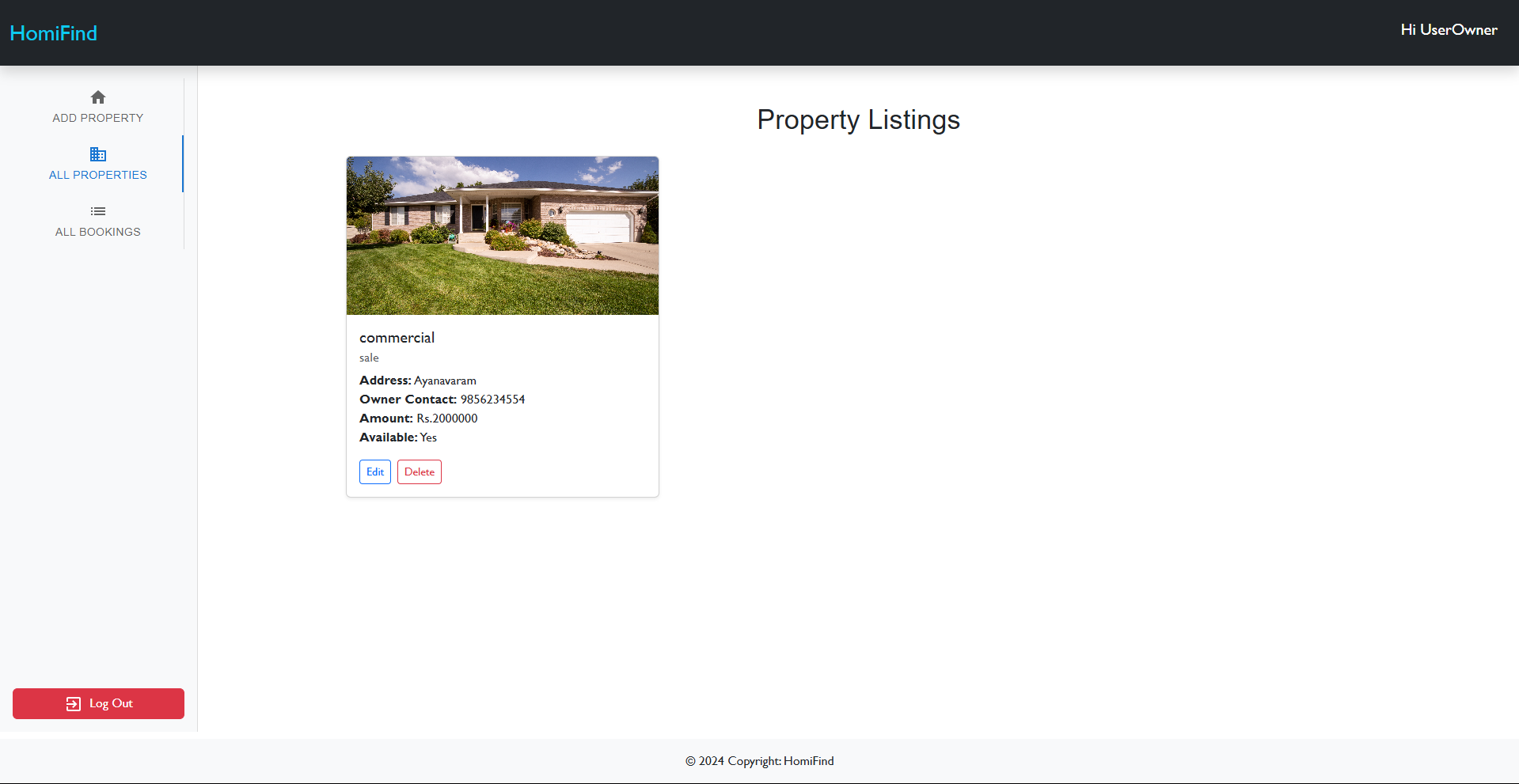
* **Admin BookingPage**



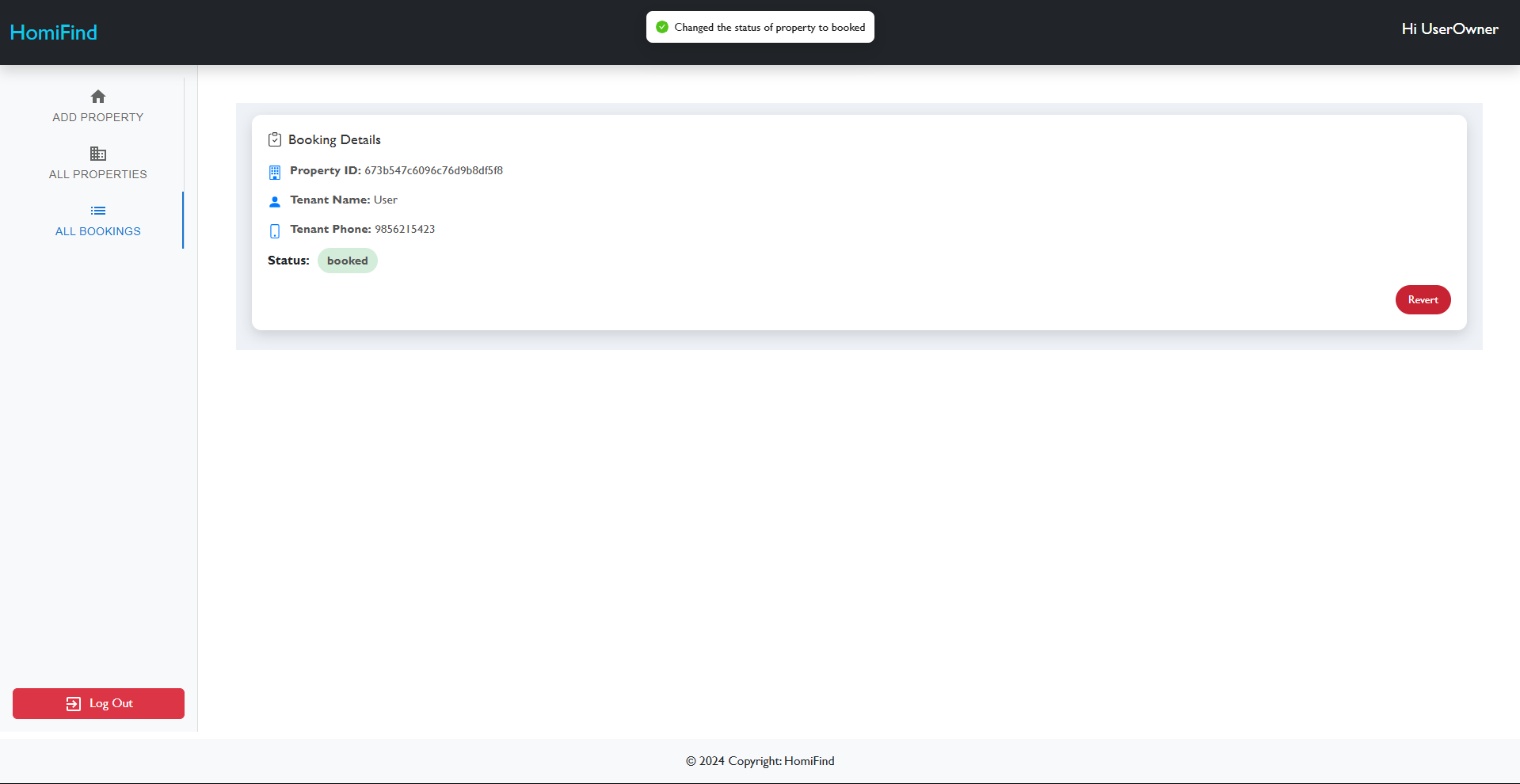
* **Owner AddProperty**



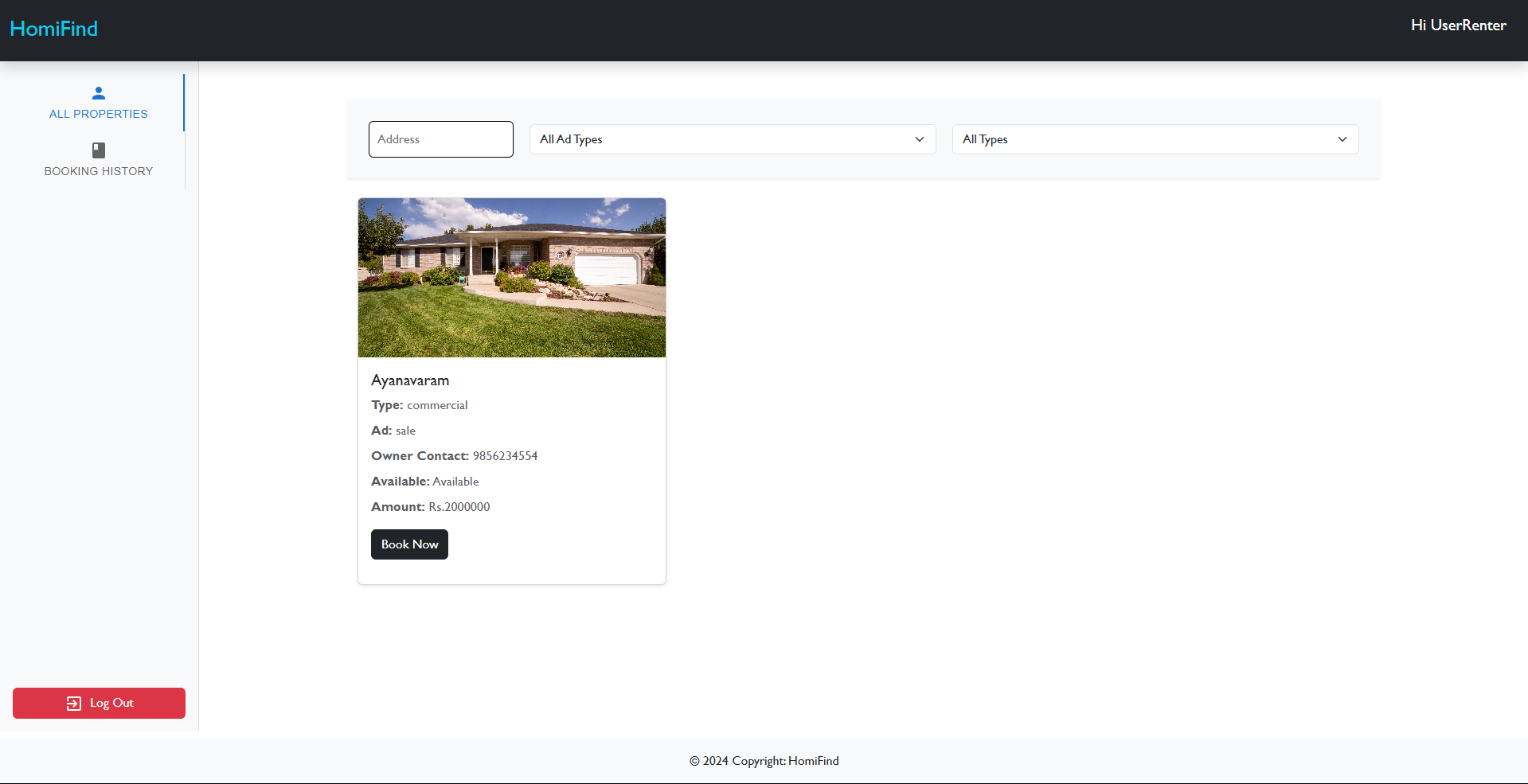
* **Owner PropertyPage**



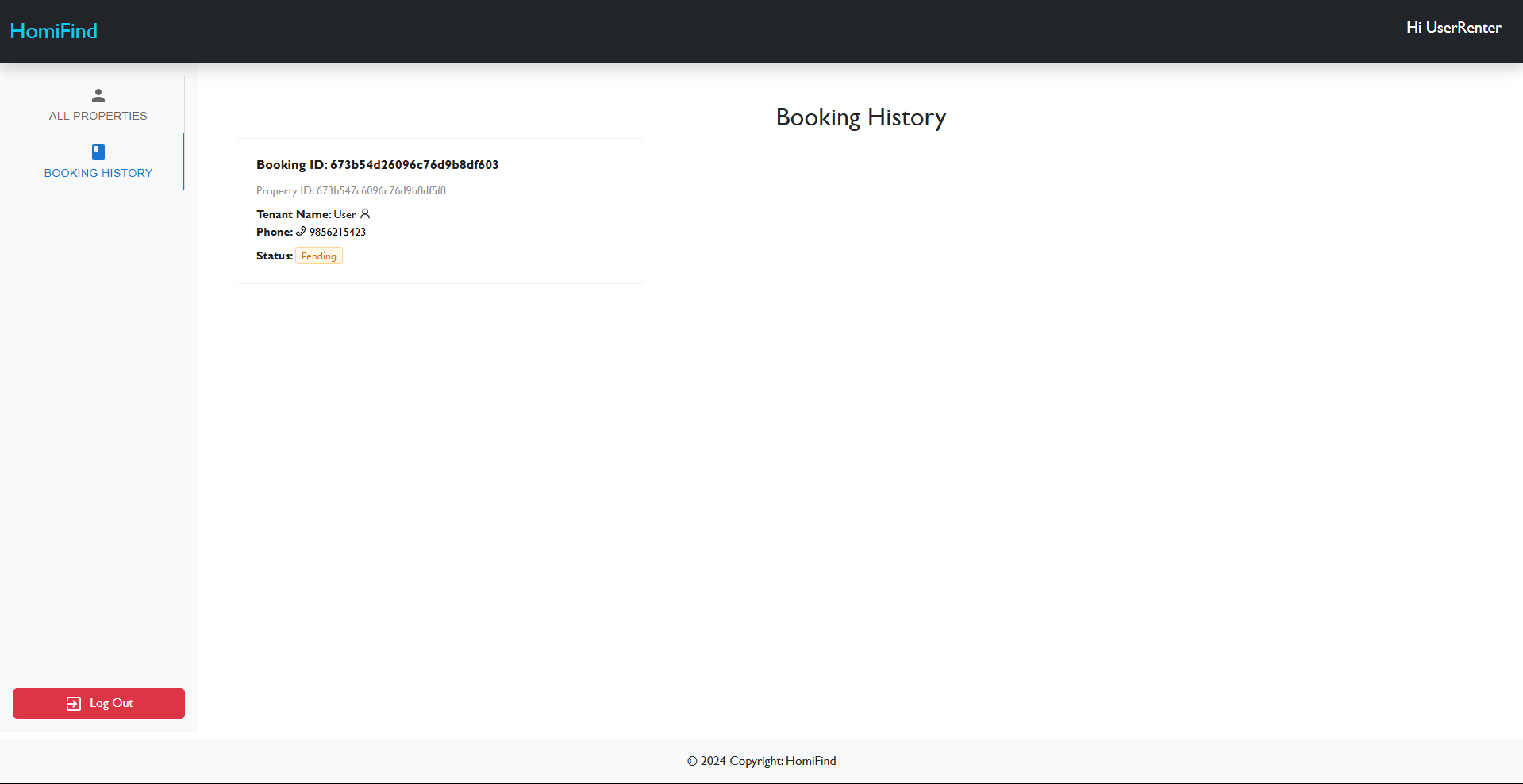
* **Owner BookingPage**



* **Renter PropertyPage**



* **Renter BookingPage**



## X. Testing

- **Unit Testing:** Covers essential functionalities and API endpoints.  
- Tools: Jest, Mocha.

## XI. Demo & Screenshots

- Demo Link: [Cick Here](https://drive.google.com/file/d/1jmajUh4hAs6fWk201yIM0ObepgmrzlfV/view?usp=drive_link)

## XII. Known Issues

* **Authentication & Security:** Implementing JWT for secure authentication while ensuring proper session handling was crucial for the system’s integrity.
* **Property Filtering:** Implementing dynamic property filtering by different categories such as location, price and type, while ensuring that the data loads efficiently.
* **Admin Management**: Creating an intuitive admin dashboard for managing user registrations and booking and ensuring it is scalable.

## XIII. Future Enhancements

* **Payment Gateway Integration:** To facilitate online payments for property bookings.
* **Advanced Search Filters:** Add more filtering options such as square footage, amenities etc.
* **Real-Time Notifications:** Notify user of booking confirmations or status changes via email or in-app m=notifications.