**DOCUMENTATION OF HOUSE RENT APP USING MERN**

1. **Introduction:**

**Project Title:** House Rent App Using MERN

**Team Members:**

* Karthiga .G
* Nathiya .R
* Nivetha .A
* Jeni Vinoliya .A
* Punitha .M

1. **Project Overview:**

* **Purpose:**

The house rent app is designed to provide an online platform where users can browse, search, and list rental properties easily. It simplifies the communication between landlords and tenants.

* **Features:**
* **User Authentication:**

Login and registration functionality for users (landlords and tenants).

* **Property Listing:**

Landlords can upload property details,

including images, rent, and location.

* **Search Functionality:**

Tenants can filter properties by price, location and amenities.

1. **Architecture:**

* Frontend:

Built using React for the UI, with routing handled by React Router.

* Backend:

Developed using Node.js and Express.js to create Restful APIs.

* Database:

MongoDb stores the user details, property listings, and transaction records.

1. **Setup Instructions:**

* Prerequisites:
  + - 1. Install Node.js(v16+)
      2. MongoDB(v5+)
      3. Install npm as yarn package manager
* Installation Steps:

1. Clone the repository [repository – link].
2. Navigate to the root directory.
3. Install dependencies for both frontend and backend.

Backend: [ cd server && npm install]

Frontend: [cd server && npm install]

1. Set up .env files for the backend.

**5.Folder Structure:**

* **Frontend:**

1. ‘Public/’ : Displays the consistent base structure for the app.
2. ‘Src/’ - core logic and UI of the app.

Components – (Navbar, Footer, and propertyCard)

Pages – (Home, Login and Register)

Services – (Authentication and fetching properties)

Styles – (CSS for PropertyCard)

1. .gitignore – prevent unnecessary files from being committed.
2. Package.json – manages dependencies

* **Backend:**

* **Config/ -** Holds the configuration files
* **Controllers/ -** Implement logic for handling API requests.
* **Middlewares/ -** Contains functions for API request processing
* **Routes/ -** Handling the requests
* **Models/ -** Defining the data structures of user data, property details.
* **Uploads/ -** Handles the images upload for property listings.

**6.Running the Application:**

* **Frontend:** Run npm start in the client directory.
* **Backend:** Run npm start in the server directory.

**7. API Documentation:**

**1. User Registration:**

**Method:** POST

**Description:** Register a new user (tenant or landlord**).**

**2. User Login:**

**Method:** POST

**Description:** Authenticates the user.

**3.Create Property:**

**Method:** POST

**Description:** Adds a new property (landlords only).

**4.Get all Properties:**

**Method:** GET

**Description:** Fetches all available properties.

**5.Get Property by Id:**

**Method:** GET

**Description:** Fetches details of specific property by ID.

**6.Update Property:**

**Method:** PUT

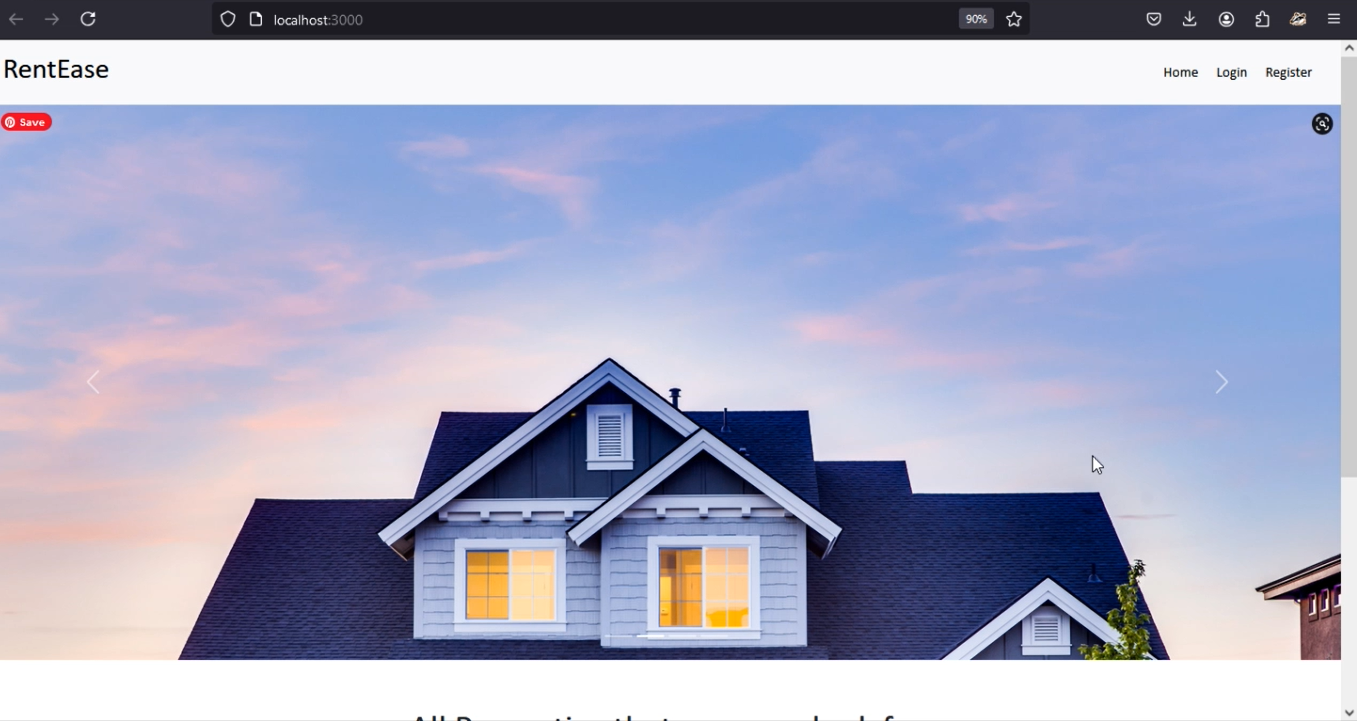
**Description:** Update details of a property (landlords only).

**8.Authentication:**

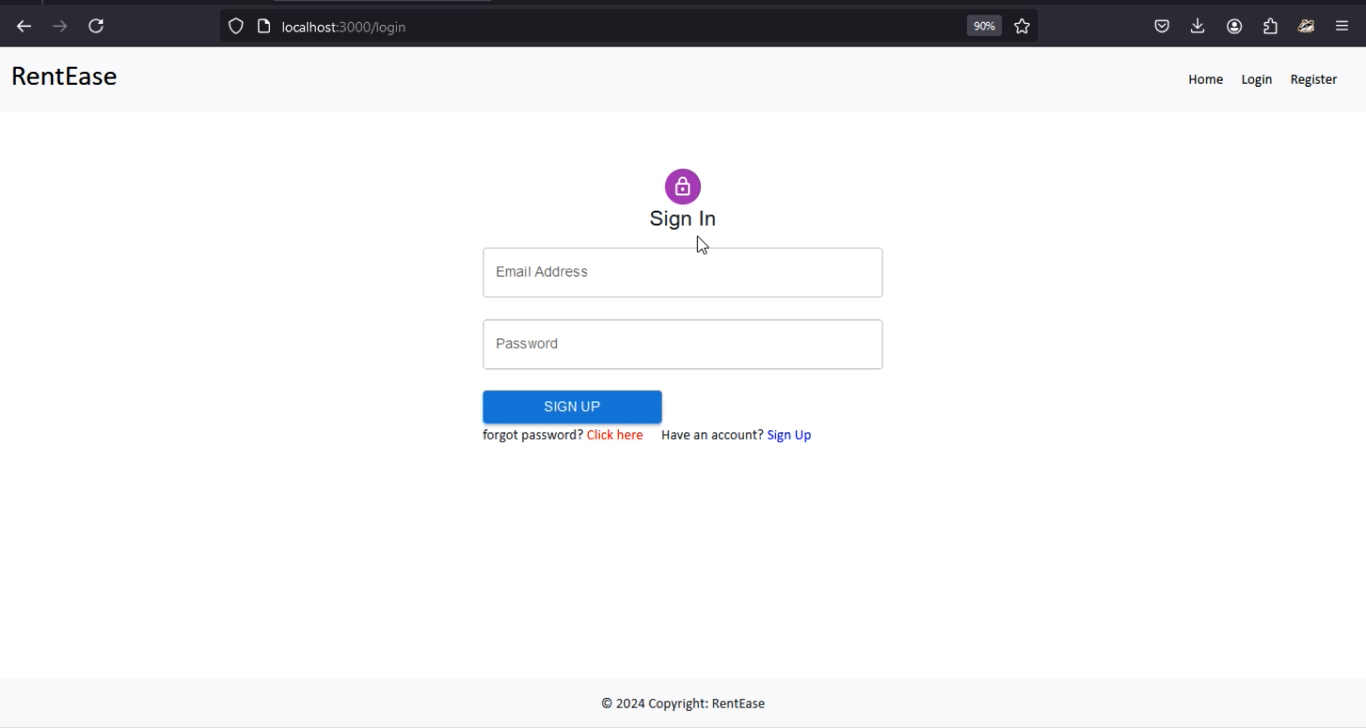
verifying users through email/phone registration and verification (via OTP or email), followed by login using credentials or social media accounts.successful login, a session token (JWT) is issued for secure access to app features, with expiration times to enhance security. Role-based access controls ensure that different user types (e.g., tenants and admins) have appropriate permissions.

**9. User Interface:**

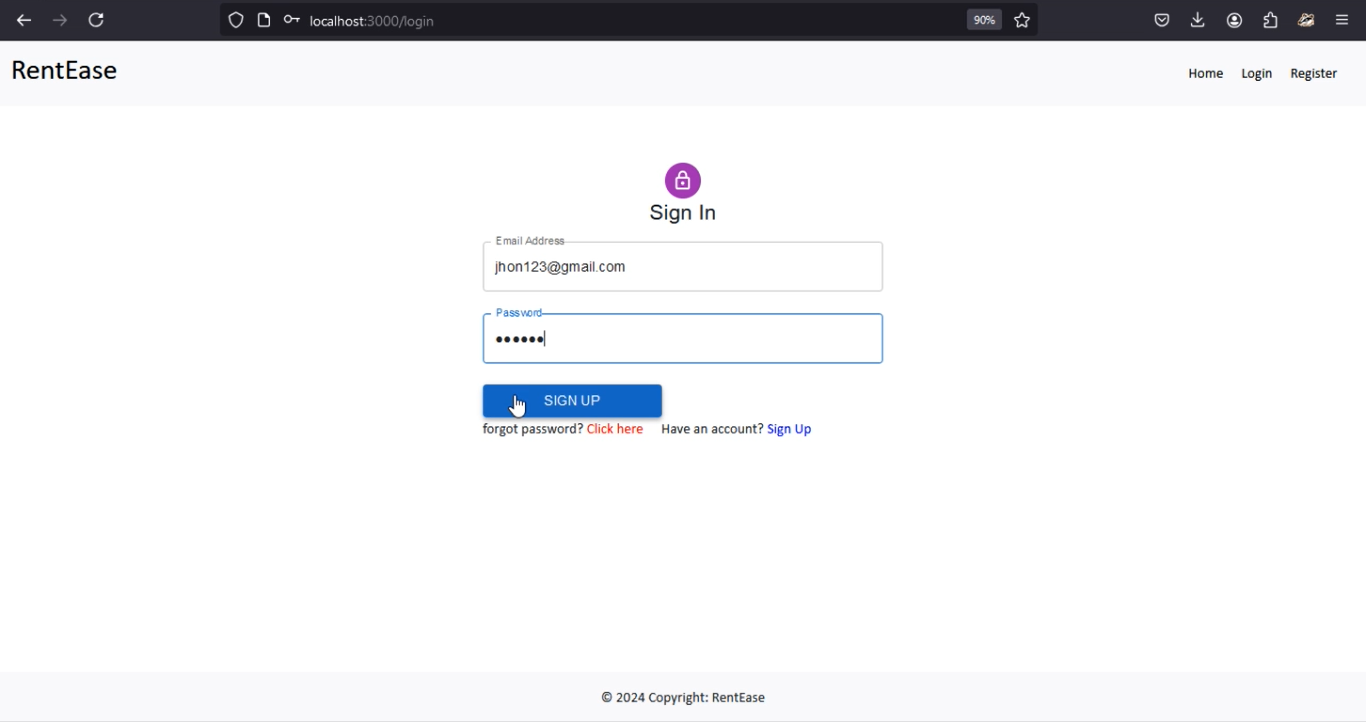
**Home Page**

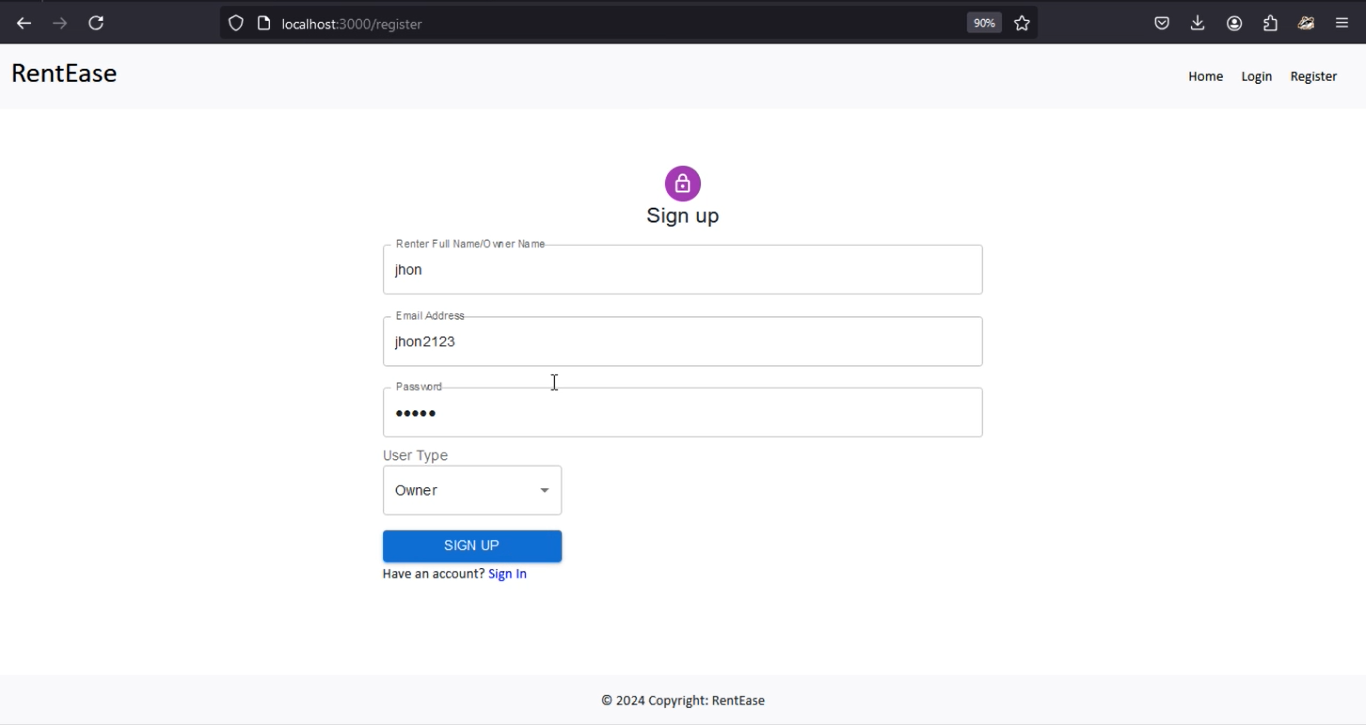
****

**User Login**

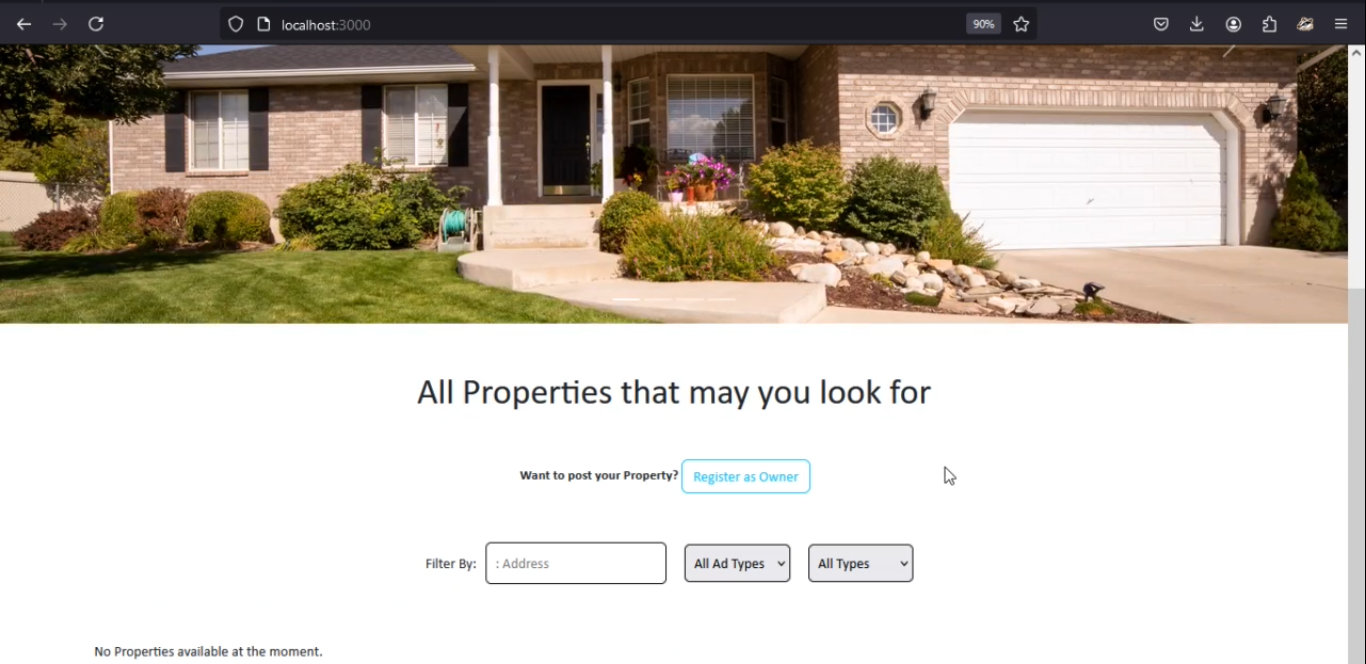
****

**User Signup**

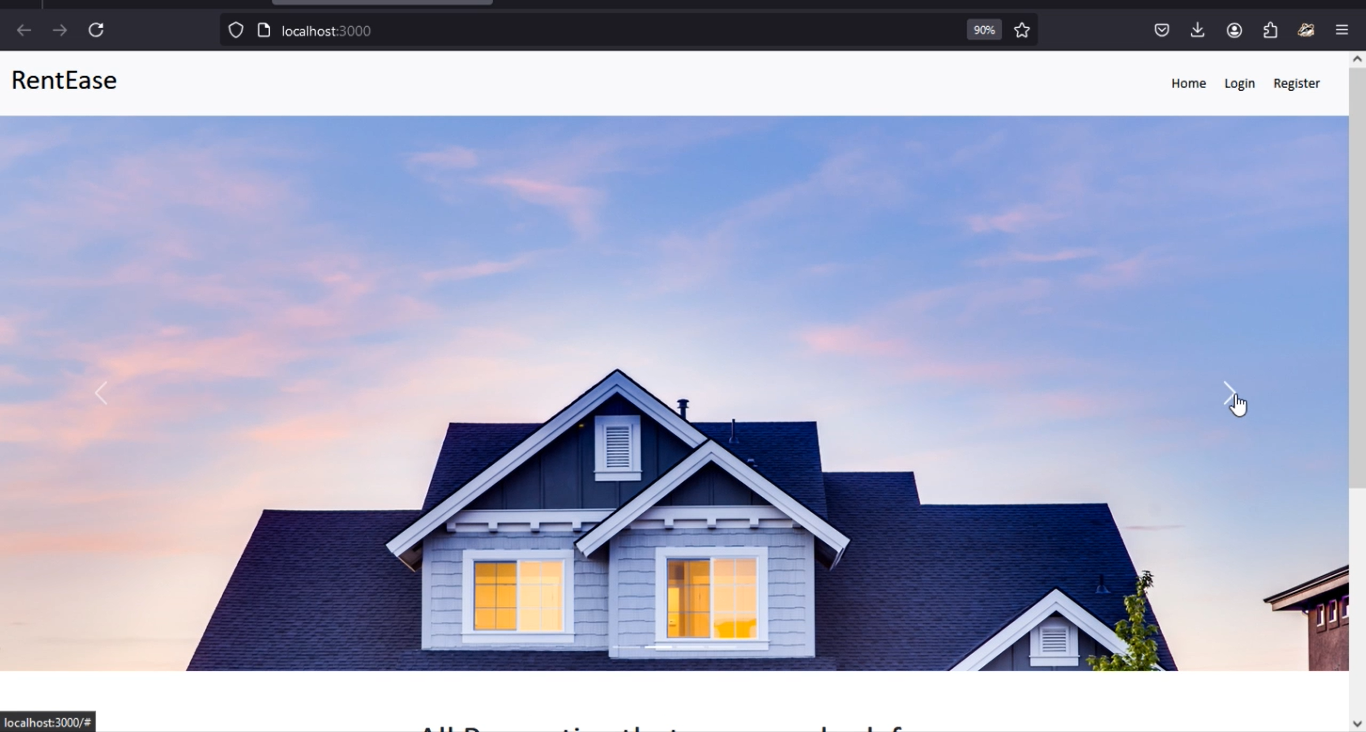
**  
 Owner Login page**

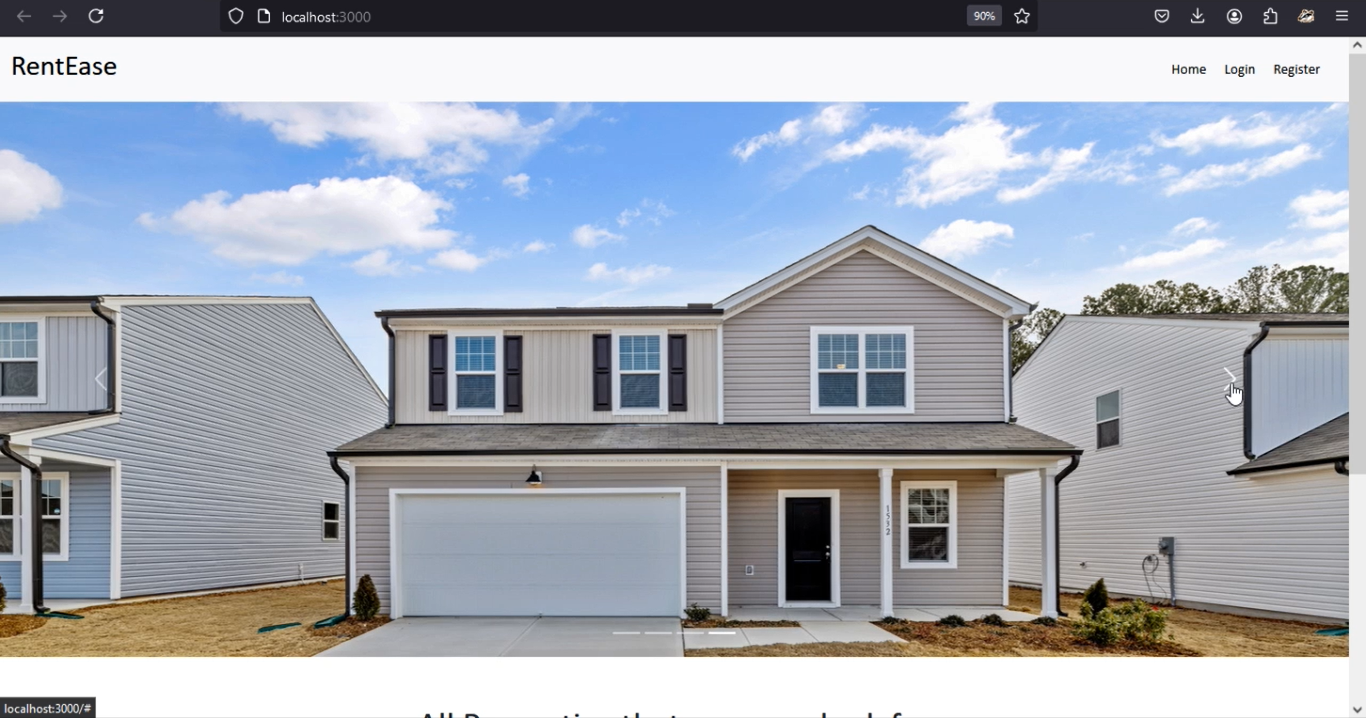
****

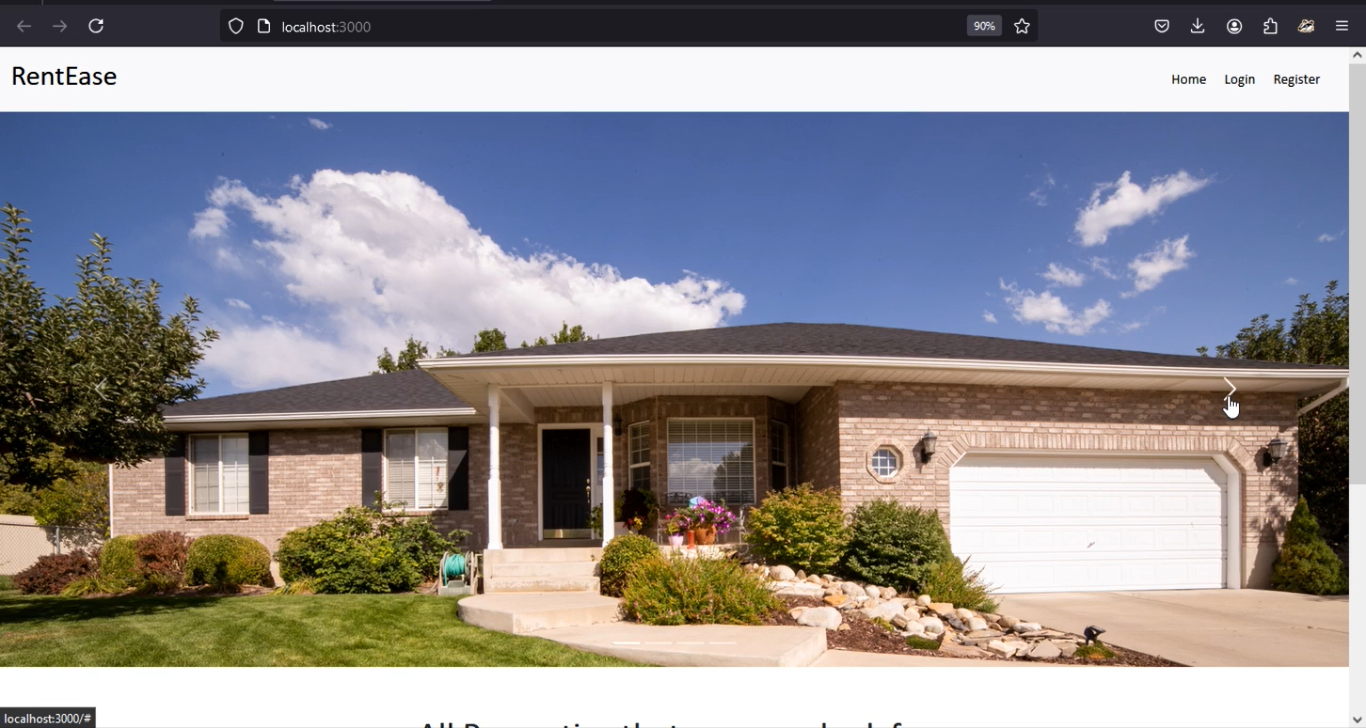
**Register as owner page**

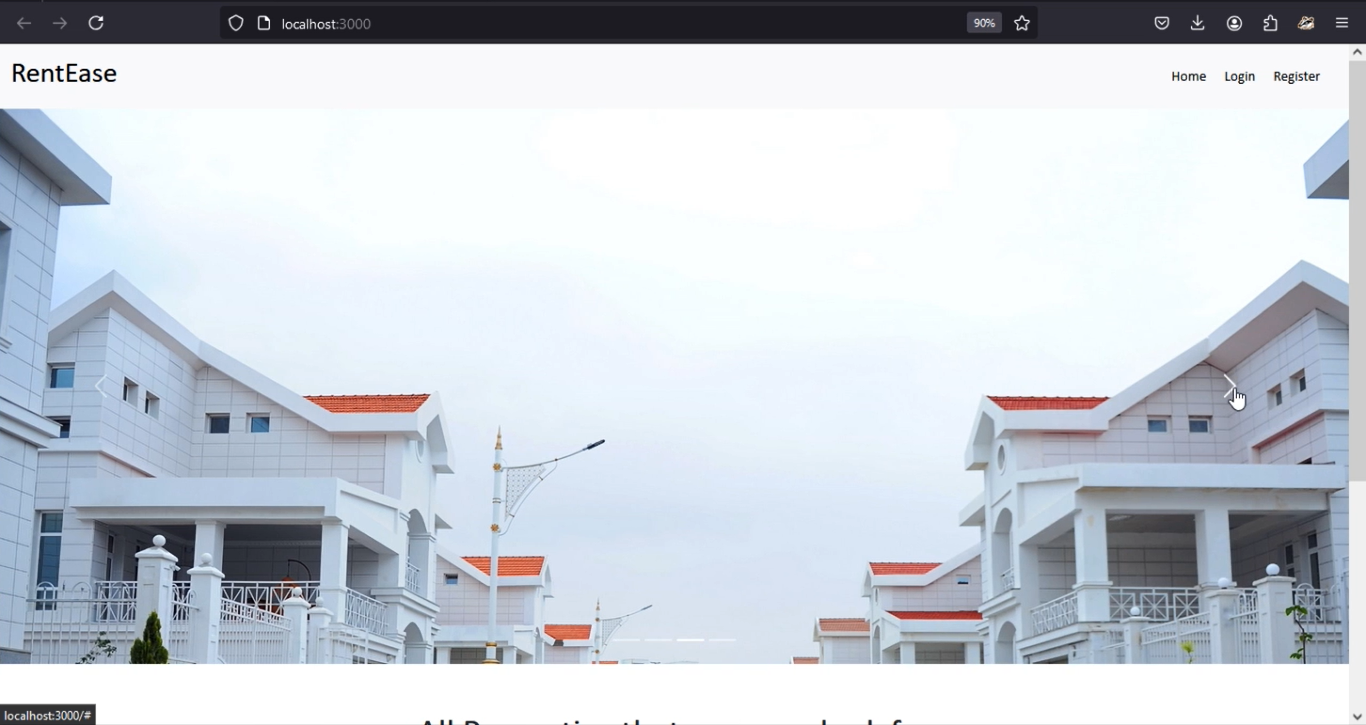
****

**Property for Rent in App**

****

****

****

****

**10. Testing:**

* API Testing checks the endpoints return the correct data and status codes.
* Unit Testing checks the models, controllers, and utility functions.
* Integration Testing checks component interact with the API and the state management.
* Security Testing checks the tools to identify the security vulnerabilities.

**12. Known Issues:**

* Issues with integrating payment gateways or handling transaction statues can lead to fail payments
* Incorrect bookings
* Double bookings

Affect user experience and trust

**13. Future Enhancement:**

* Location Based suggestions to recommend properties based on the user’s current location or preferred areas
* Enable real-time messaging between tenants and the owner for quicker responses and better communication
* Notify users about new listings, booking updates, price drops, or upcoming rental deadlines.