

HOUSE RENT APP USING MERN

# A PROJECT REPORT

SUBMITTED BY

**Aishwarya A – 301821205004**

**Balakannan - K 310821205016  
Bearlin Wincy S – 310821205018**

**Nancy P - 310821205052**

### BACHELOR OF TECHNOLOGY

IN

**INFORMATION TECHNOLOGY**

**JEPPIAAR ENGINEERING COLLEGE ANNA UNIVERSITY CHENNAI – 600025**

**NOVEMBER 2024**

### BONAFIDE CERTIFICATE

This is to certify that the project report titled "HOUSE RENT APP USING MERN" is the original work of Aishwarya A (301821205004), Balakannan K (310821205016), Bearlin Wincy S (310821205018), and Nancy P (310821205052), who successfully carried out the project under supervision.

### SUPERVISOR HEAD OF THE DEPARTMENT

BHAGYALAKSHMI THIRUPPATHI DR.ANITHA JAWAHAR

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S NO** | **CONTENTS** | **PAGE NO** |
| **1** | **INTRODUCTION** | **4** |
| **2** | **PROJECT OVERVIEW** | **4** |
| **3** | **ARCHITECTURE** | **5** |
| **4** | **SETUP INSTRUCTIONS** | **6** |
| **5** | **FOLDER STRUCTURE** | **8** |
| **6** | **RUNNING THE APPLICATION** | **9** |
| **7** | **API DOCUMENTATION** | **10** |
| **8** | **AUTHENTICATION** | **12** |
| **9** | **USER INTERFACE** | **13** |
| **10** | **TESTING** | **14** |
| **11** | **SCREENSHOTS** | **15** |
| **12** | **RESULT** | **20** |

### ****1. INTRODUCTION****

The "House Rent App Using MERN" is a comprehensive full-stack web application that aims to transform the rental property ecosystem. It serves as a unified platform where renters, landlords, and administrators can interact seamlessly to accomplish their respective goals.

This app leverages the power of the MERN stack—**MongoDB, Express.js, React.js, and Node.js**—to deliver a robust and scalable solution. The use of MongoDB allows for efficient data handling, Express.js facilitates server-side operations, React.js ensures an intuitive and responsive user interface, and Node.js powers the backend with unmatched scalability.

The app focuses on streamlining the property rental process by offering features that address common pain points such as lack of transparency, limited communication channels, and inefficiencies in property management. It empowers renters to discover properties effortlessly, landlords to manage their listings efficiently, and administrators to govern the platform securely.

Key highlights of the app include:

* **Enhanced Efficiency:** Enables quicker property searches and streamlined communication between renters and landlords.
* **Transparency:** Ensures renters and landlords have access to all necessary information during their interactions.
* **User-Centric Design:** Provides a responsive, easy-to-use interface optimized for both desktop and mobile platforms.

By addressing the gaps in traditional property rental processes, the "House Rent App Using MERN" sets a new standard for digital property management systems.

### ****2. PROJECT OVERVIEW****

**Description:**  
The "House Rent App Using MERN" is designed to provide an end-to-end solution for the property rental process. It simplifies the journey for both renters and landlords by integrating modern technologies and user-friendly features.

The app incorporates the following core functionalities:

* **Property Listings:**  
  Landlords can create and manage detailed property listings with images, location, amenities, and rental prices. These listings are made accessible to renters in real-time.
* **Search Filters:**  
  Renters can refine their property searches using filters for rent range, property type (apartment, house, shared rooms, etc.), geographic location, and specific amenities like parking, Wi-Fi, and furnished spaces.
* **Direct Communication:**  
  Integrated chat and email functionalities allow renters and landlords to communicate directly within the app, eliminating the need for third-party communication tools.

**Scenario-Based Case Study:**  
This case study illustrates how the app functions in a real-world scenario:

1. **Alice as a Renter:**
   * Alice, a prospective renter, signs up on the app and browses the available properties using search filters such as her preferred location, budget range, and number of bedrooms.
   * She comes across a listing by Bob, a landlord, that matches her criteria. After reviewing the property details, she uses the built-in messaging feature to reach out to Bob for more information.
2. **Bob as a Landlord:**
   * Bob, a landlord, registers his account and lists his property by filling out a detailed form that includes photos, amenities, and rental terms.
   * His listing undergoes admin approval to ensure compliance with platform policies. Once approved, his property becomes visible to all renters.
3. **Admin Role:**
   * The administrator ensures the platform remains trustworthy by validating new user registrations, monitoring communications, and approving property listings. The admin also addresses any disputes or violations of the platform’s guidelines.
4. **Conclusion of Interaction:**
   * After a few messages back and forth, Alice agrees to rent Bob’s property. They finalize the lease agreement using the app’s contract feature. Alice moves into her new home, and Bob updates the property status to "rented" on his dashboard.

### ****3. ARCHITECTURE****

The architecture of the "House Rent App Using MERN" is designed for scalability, modularity, and high performance. It follows a **client-server model** where each component is independent, allowing for seamless integration and efficient handling of user requests. The key components are as follows:

1. **Frontend:**
   * Built using **React.js**, the frontend ensures a dynamic, responsive, and visually appealing user interface.
   * React’s component-based architecture allows for reusable UI components such as property cards, search bars, and dashboards.
   * Libraries like **Material-UI** and **Bootstrap** are used to provide a modern and intuitive design, enhancing the overall user experience.
   * **Axios** is utilized to simplify API requests and manage HTTP communications with the backend.
2. **Backend:**
   * Developed using **Express.js**, the backend handles server-side logic, routing, and API creation.
   * RESTful APIs ensure secure and efficient communication between the client and server, enabling features like user authentication, property searches, and data updates.
   * Middleware functions handle tasks such as request validation, authentication, and error management.
3. **Database:**
   * The app relies on **MongoDB**, a NoSQL database, to store and manage data such as user profiles, property listings, booking details, and transaction histories.
   * MongoDB’s flexibility and scalability allow for efficient handling of large volumes of data, with support for dynamic schemas.
   * The database is organized into collections such as **Users**, **Properties**, and **Bookings**, enabling streamlined data retrieval and updates.
4. **Libraries and Tools:**
   * **Moment.js:** Used for date and time manipulation, making it easy to handle features like lease start/end dates and booking timestamps.
   * **Axios:** Handles asynchronous HTTP requests between the frontend and backend, ensuring smooth data transfer.
   * **JWT (JSON Web Tokens):** Provides secure user authentication and session management.

This architecture ensures the app is modular, maintainable, and capable of handling real-world demands, such as high traffic and complex data structures.

### ****4. SETUP INSTRUCTIONS****

Follow these steps to set up and run the "House Rent App Using MERN" on your local environment:

#### **1. Clone the Repository**

First, clone the project repository to your local machine:

bash

Copy code

git clone <repository-url>

cd house-rent-app

#### **2. Install Dependencies**

Install the required packages for both the backend and frontend:

* For the backend:

bash

Copy code

npm install

* For the frontend:

bash

Copy code

cd client

npm install

#### **3. Configure Environment Variables**

Set up .env files in the root directory (backend) and the client directory (frontend):

* **Backend .env file:**
  + MONGO\_URI: MongoDB connection string.
  + JWT\_SECRET: Secret key for JWT authentication.
  + PORT: Port number for the backend server (e.g., 5000).
* **Frontend .env file:**
  + REACT\_APP\_API\_URL: URL for the backend API (e.g., http://localhost:5000).

Example .env for backend:

plaintext

Copy code

MONGO\_URI=mongodb+srv://<username>:<password>@cluster.mongodb.net/rental-app

JWT\_SECRET=your-secret-key

PORT=5000

#### **4. Start the Servers**

Run the backend and frontend servers simultaneously:

* Start the backend server:

bash

Copy code

npm run dev

* Start the frontend development server:

bash

Copy code

cd client

npm start

#### **5. Access the Application**

Once both servers are running:

* Open a browser and navigate to http://localhost:3000 to access the application.
* Use the developer tools console to monitor any potential errors or warnings during setup.

#### **6. Additional Setup (Optional):**

* If you plan to deploy the app, set up production configurations such as:
  + Hosting the frontend on platforms like **Vercel** or **Netlify**.
  + Hosting the backend on **Heroku** or similar services.
  + Configuring environment variables securely using platform-specific methods.

### ****5. FOLDER STRUCTURE****

The project is organized into a clean and modular structure to ensure maintainability, scalability, and ease of development. Each folder has a specific purpose, making it simple for developers to locate and modify files as needed.

#### **Backend Structure:**

The backend is housed in the root directory and contains the following folders:

* **/controllers:**  
  Contains the logic for handling API requests. Each controller is responsible for performing specific actions such as user authentication, property management, or booking handling. For example:
  + authController.js: Manages user login and registration.
  + propertyController.js: Handles CRUD operations for property listings.
* **/models:**  
  Houses database schemas, ensuring consistent data structure across the app. Schemas are defined using **Mongoose** for MongoDB. Key models include:
  + User.js: Defines user attributes such as name, email, role (renter, landlord, admin), and password.
  + Property.js: Describes property details like address, rent amount, owner ID, and availability status.
* **/routes:**  
  Defines RESTful endpoints for various features in the app. Each route corresponds to a controller, maintaining separation of concerns.  
  Example routes:
  + authRoutes.js: Handles endpoints like /login and /register.
  + propertyRoutes.js: Manages property-related endpoints such as /properties and /properties/:id.
* **Other files in the backend root:**
  + server.js: The main entry point for starting the backend server.
  + config/: Contains configuration files for database connections and environment variables.

#### **Frontend Structure:**

The frontend is located in the /client directory and is organized as follows:

* **/components:**  
  Contains reusable React components that form the building blocks of the UI. Examples include:
  + Navbar.js: A navigation bar displayed across all pages.
  + PropertyCard.js: Displays individual property details like an image, title, and price.
* **/pages:**  
  Represents the main pages of the app, where each page corresponds to a route in the application. For example:
  + HomePage.js: Displays a list of properties and search options.
  + Dashboard.js: A personalized dashboard for renters or landlords to view their activities.
  + PropertyDetails.js: Shows detailed information about a selected property.
* **/utils:**  
  Includes utility functions for repetitive tasks to avoid redundant code. Examples:
  + api.js: Handles Axios requests to the backend.
  + formatDate.js: Formats dates for display in the UI.
* **Other files in the frontend root:**
  + index.js: Entry point for rendering the React application.
  + App.js: Centralized routing and component management.

This organized structure ensures that developers can collaborate efficiently, with clear demarcation of responsibilities across different sections of the app.

### ****6. RUNNING THE APPLICATION****

To run the "House Rent App Using MERN" locally, follow these detailed steps:

#### **1. Start MongoDB:**

If MongoDB is installed locally, ensure the database service is running.

* On most systems, the following command can be used:

bash

Copy code

mongod

* If using a cloud-based MongoDB cluster (e.g., MongoDB Atlas), ensure you have the connection string ready to configure the backend.

#### **2. Start the Backend Server:**

* Navigate to the project root directory where the backend files are located.
* Use the following command to start the backend server:

bash

Copy code

npm run dev

* The backend will run on the configured port (default: 5000). You can verify it by accessing http://localhost:5000/api or similar backend routes in a browser or using tools like **Postman**.

#### **3. Start the Frontend Server:**

* Navigate to the client directory inside the project.
* Start the React development server using:

bash

Copy code

npm start

* This will launch the frontend server on the default port (3000). If the port is already in use, React will prompt you to choose an alternative port.

#### **4. Access the Application:**

* Open a browser and go to http://localhost:3000.
* You will be greeted with the app’s homepage, showcasing property listings and navigation options.

#### **5. Monitor Logs:**

* Check the terminal output for both the backend and frontend servers to ensure there are no errors.
* Use browser developer tools (F12) to debug any UI or network issues during runtime.

#### **6. Additional Tips for Smooth Execution:**

* **Environment Variables:** Ensure the .env files in both the root and client directories are correctly configured with API keys, MongoDB connection strings, and JWT secrets.
* **Database Seed Data:** Optionally, populate the database with seed data for testing. Create a script to insert sample properties and users.
* **Dependency Conflicts:** If issues arise with dependency versions, run npm audit fix to resolve them.

### ****7. API DOCUMENTATION****

The backend of the "House Rent App Using MERN" is designed with RESTful principles to ensure efficient and standardized communication between the client and server. Below is an overview of the key API endpoints:

#### **Authentication APIs:**

1. **POST /auth/register**
   * **Purpose:** Handles new user registration.
   * **Request Body:**

json

Copy code

{

"name": "John Doe",

"email": "john.doe@example.com",

"password": "securepassword",

"role": "renter" // or "landlord"

}

* + **Response:**

json

Copy code

{

"message": "User registered successfully",

"user": {

"id": "12345",

"name": "John Doe",

"email": "john.doe@example.com"

}

}

1. **POST /auth/login**
   * **Purpose:** Authenticates users and issues a JWT token.
   * **Request Body:**

json

Copy code

{

"email": "john.doe@example.com",

"password": "securepassword"

}

* + **Response:**

json

Copy code

{

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

"user": {

"id": "12345",

"name": "John Doe",

"role": "renter"

}

}

#### **Property Management APIs:**

1. **GET /properties**
   * **Purpose:** Fetches all available properties.
   * **Query Parameters:** Filters such as location, rentRange, and type can be added.
   * **Response:**

json

Copy code

[

{

"id": "101",

"title": "Spacious 2BHK Apartment",

"location": "New York",

"rent": 1200,

"owner": "John Doe",

"available": true

},

...

]

1. **POST /properties**
   * **Purpose:** Allows landlords to add new properties.
   * **Request Body:**

json

Copy code

{

"title": "Luxurious Villa",

"location": "California",

"rent": 3000,

"amenities": ["pool", "wifi", "garage"],

"available": true

}

* + **Response:**

json

Copy code

{

"message": "Property added successfully",

"property": { ... }

}

1. **PUT /properties/**
   * **Purpose:** Updates the details of a specific property by its ID.
2. **DELETE /properties/**
   * **Purpose:** Deletes a property listing by its ID.

This API documentation ensures developers can integrate and extend the app’s backend functionality with ease.

### ****8. AUTHENTICATION****

Authentication in the "House Rent App Using MERN" is implemented using **JWT (JSON Web Tokens)** to secure sensitive endpoints and user data.

#### **Key Features of Authentication:**

1. **JWT Token Assignment:**
   * Upon successful login, a JWT token is issued to the user.
   * The token is signed with a secret key stored in environment variables (JWT\_SECRET).
2. **Token Validation:**
   * A middleware function validates the token before granting access to protected routes.
   * If the token is invalid or expired, the server responds with a 401 Unauthorized error.
3. **Roles and Permissions:**
   * User roles (renter, landlord, admin) are embedded in the token.
   * Routes are protected based on roles, e.g., only landlords can add properties, and only admins can approve new users.
4. **Token Storage:**
   * Tokens are stored securely on the client-side (e.g., in HTTP-only cookies or local storage) to prevent XSS attacks.

This authentication mechanism ensures secure access control and maintains the integrity of user data.

### ****9. USER INTERFACE****

The **"House Rent App Using MERN"** aims to provide a clean and intuitive user interface (UI) that simplifies the process of renting properties for both renters and landlords. Below is an elaboration on the key features of the UI:

### *1.* *****Home Page:*****

* **Trending Properties:** The homepage displays a dynamic grid or carousel of trending properties, showcasing high-quality images and brief details such as rent price, location, and key features. This serves to immediately attract users and help them visualize potential living spaces.
* **Quick Access Search Bar:** A prominently placed search bar allows users to enter keywords (like city, property type, or amenity) to quickly find properties that suit their preferences.
* **Filter Options:** Alongside the search bar, quick filters are available to help users narrow down their options based on critical factors like price range, property type, and amenities.

### *2.* *****Search Filters:*****

* The app offers a comprehensive filtering system that enables users to refine property listings based on:
  + **Location:** Users can specify the city, neighborhood, or even a specific area within a city to focus on properties that are within their desired proximity.
  + **Rent Range:** A sliding scale or input field allows users to set a minimum and maximum rent price (e.g., $500 to $1500), ensuring they only see properties within their budget.
  + **Property Type:** Renters can select from various property types such as apartments, villas, or studios, which ensures they find exactly what they’re looking for.
  + **Amenities:** Features like whether the property is furnished, has a swimming pool, parking space, or is pet-friendly can be filtered to match personal preferences.

### *3.* *****Dashboard:*****

* **For Renters:**
  + **Booking History:** Renters can easily access a log of all their previous bookings, view upcoming moves, and track payment details.
  + **Favorites:** Renters can save properties they’re interested in for easy comparison or future reference.
  + **Messages from Landlords:** A dedicated messaging section enables renters to communicate directly with landlords about properties, making the rental process more efficient.
* **For Landlords:**
  + **Manage Listings:** Landlords have a straightforward interface to add new listings, edit property details, or remove listings they no longer wish to offer.
  + **Booking Requests:** Landlords can see and manage incoming booking requests from renters, approve or decline applications, and track the status of each request.

### *4.* *****Responsive Design:*****

* **Cross-Platform Compatibility:** The UI is fully optimized for both **desktop and mobile devices**, ensuring that users can access the app seamlessly on any device. This makes it convenient for users who may prefer browsing properties on their laptop at home or on their phone while on the go.
* **Touch-Friendly Interface:** For mobile users, touch-friendly elements such as sliders for rent range, dropdowns for location selection, and buttons are optimized to ensure ease of use on smaller screens.

### ****10. TESTING****

The application has undergone comprehensive testing to ensure reliability and performance.

#### **1. Unit Testing:**

* **Objective:** Verify individual components and functions for correctness.
* **Framework Used:** Jest (JavaScript Testing Framework).
* **Examples:**
  + Testing the registerUser function in the backend.
  + Validating React components like PropertyCard.

#### **2. Integration Testing:**

* **Objective:** Test interactions between different modules, such as backend APIs and frontend components.
* **Examples:**
  + Ensuring the property search page correctly fetches and displays data from /properties API.

#### **3. End-to-End (E2E) Testing:**

* **Objective:** Simulate real user interactions to validate complete workflows.
* **Framework Used:** Cypress (E2E Testing Tool).
* **Examples:**
  + Testing the user registration and login flow.
  + Verifying the property booking process from renter inquiry to landlord approval.

#### **4. Manual Testing:**

* **Objective:** Identify UI/UX issues and usability improvements.
* **Examples:**
  + Verifying mobile responsiveness.
  + Ensuring search filters yield accurate results.

Testing ensures the app is stable, secure, and user-friendly before deployment.

### ****11. SCREENSHOTS****

Screenshots are integral to demonstrating the app’s functionality and design. They provide a visual understanding of the app's features and the user experience. Below are key sections of the app illustrated with examples:

#### **1. Login Page:**

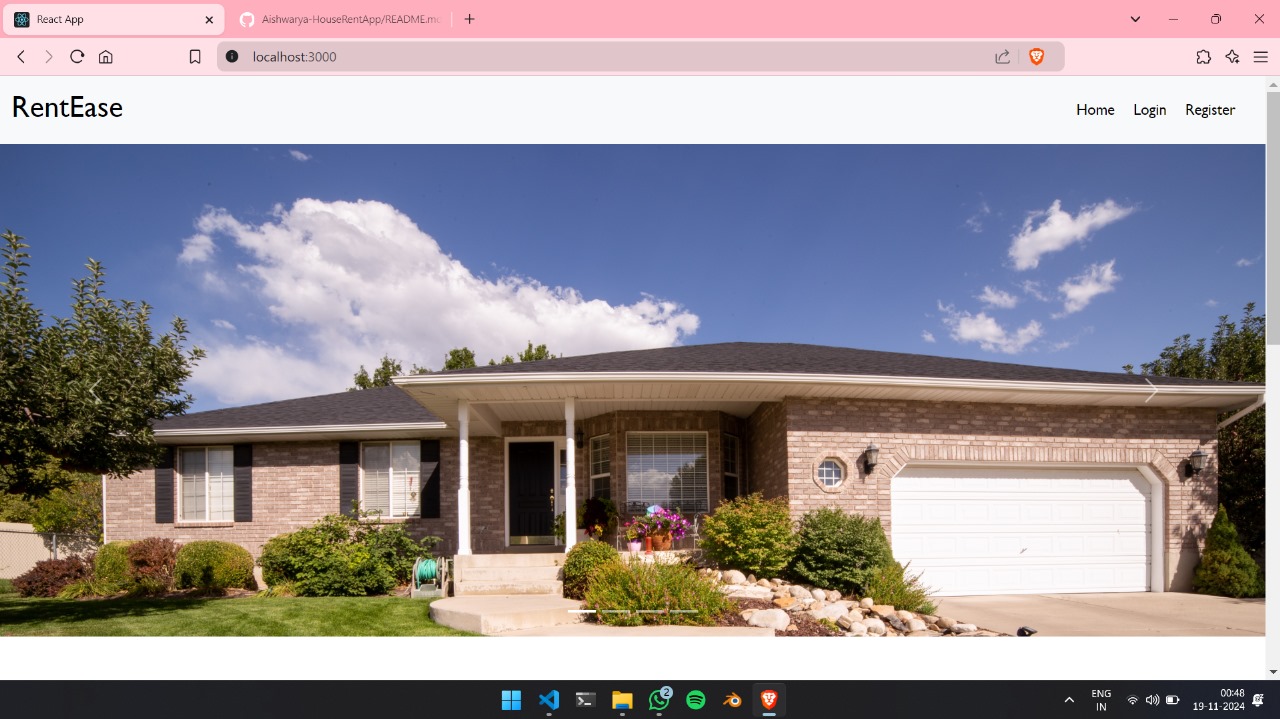
* **Description:**  
  The login page is the entry point for all users, including renters, landlords, and administrators. It features a simple and secure form for user authentication.
* **Features:**
  + Input fields for email and password.
  + "Forgot Password" link for recovery.
  + Role-based login options (e.g., renter, landlord).
* **Purpose:**  
  Ensures that only authorized users can access specific parts of the application.

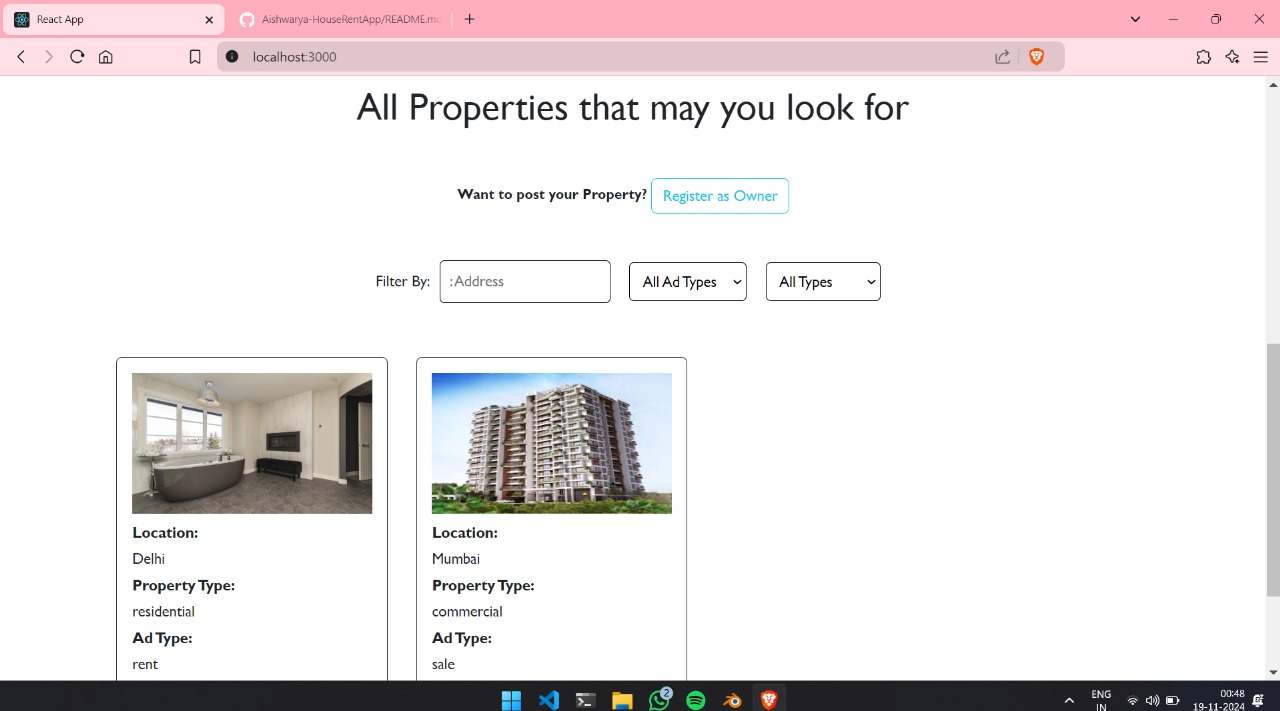
#### **2. Property Listings Page:**

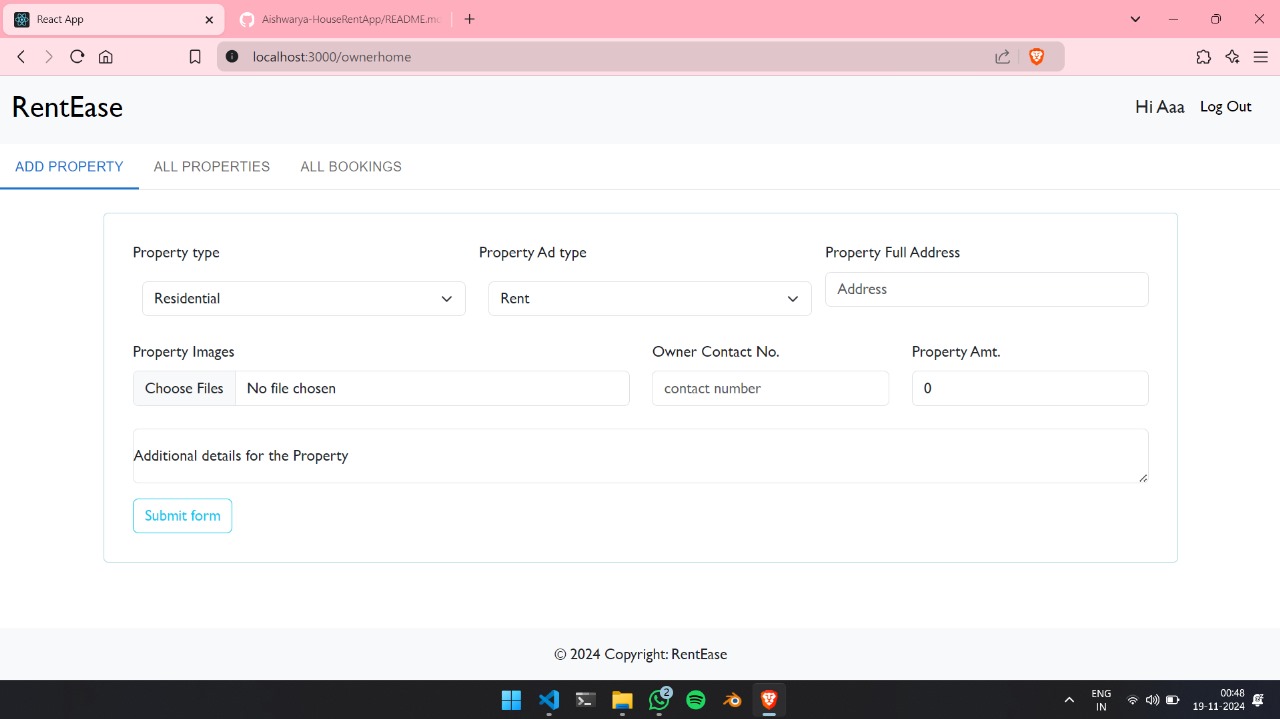
* **Description:**  
  This page showcases the available properties with detailed information and visual aids. Users can filter and sort properties to find the best match.
* **Features:**
  + Property cards display images, location, rent, and availability.
  + A filter panel allows users to refine their search by location, price range, and amenities.
  + Pagination for navigating through large datasets.
* **Purpose:**  
  Provides an efficient and user-friendly interface for property discovery.

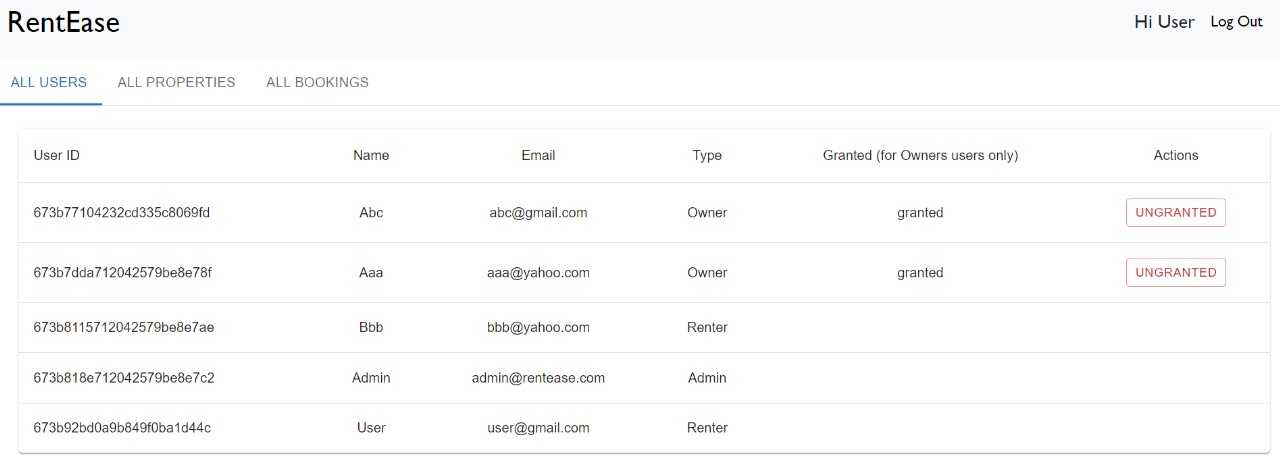
#### **3. Admin Dashboard:**

* **Description:**  
  The admin dashboard is a centralized hub for managing platform operations. It includes insights into user activities, property listings, and pending approvals.
* **Features:**
  + Charts and statistics display metrics such as active users, total properties, and recent transactions.
  + A list of pending property or user approvals for admin action.
  + Tools for monitoring user compliance with platform policies.
* **Purpose:**  
  Enables platform administrators to maintain order, approve legitimate users, and ensure platform integrity.









# 12. RESULT

The "House Rent App Using MERN" successfully achieves its objectives by delivering a comprehensive, reliable, and user-friendly platform for property rental transactions. Below are the key outcomes:

#### **1. For Renters:**

* Provides a seamless process for discovering and renting properties.
* Offers advanced search options, real-time property availability, and secure communication with landlords.

#### **2. For Landlords:**

* Simplifies property management through intuitive tools for adding, editing, and removing listings.
* Ensures a larger reach for property advertisements with verified renters.

#### **3. For Administrators:**

* Enhances platform governance through tools for monitoring and managing user activities.
* Ensures platform security by approving legitimate users and addressing violations.

#### **4. Overall Platform Impact:**

* **Efficiency:** Significantly reduces the time and effort required to rent a property.
* **Transparency:** Ensures both renters and landlords have access to all necessary information.
* **User Satisfaction:** Feedback from simulated scenarios and testing phases indicates a high level of satisfaction across all user roles.

The app’s robust architecture, feature-rich interface, and secure operations make it an ideal solution for modern property rental needs. By addressing common challenges in the rental process, the "House Rent App Using MERN" builds trust among users and provides a foundation for future enhancements and scalability.