**Naan Mudhalvan Project**

**MONGODB With MERN STACK**

**Project Title: House Rent App**

**Team Members:**

**Akshaya G K**

**AI & DS (FINAL YEAR)**

**212421243003**

**SREE SASTHA INSTITUTE OF**

**ENGINEERING AND TECHNOLOGY**

**CHEMBARAMBAKKAM**

**CHENNAI - 6000123**

**Index**

* **Project Overview**
* **Objectives**
* **Technology Stack**
* **System Requirements**
* **Features**
* **Project Architecture**
* **Installation and Setup**
* **Workflow and Usage**
* **Folder Structure**
* **ER Diagram**
* **Challenges Faced**
* **Future Enhancements**
* **Project Implementation & Execution**
* **Conclusion**

**1. Project Overview**

The House Rent App is a MERN-based web application aimed at streamlining the property rental process for both renters and property owners. This application allows renters to browse listings, inquire about properties, and secure rentals, while enabling property owners to manage their listings and interact with potential renters. Admin oversight ensures a secure, well-governed platform, maintaining quality and security for all users.

**2. Objectives**

The primary objectives of the House Rent App are to:

1. Simplify the property rental process by providing a centralized platform for renters and property owners.
2. Allow property owners to effectively manage their rental listings, respond to inquiries, and track bookings.
3. Provide renters with easy access to browse and filter properties according to specific needs.
4. Introduce admin functionality to oversee activities, validate owner accounts, and monitor the application for a safe user experience.

**3. Technology Stack**

The House Rent App is built on the MERN stack, which includes:

* **MongoDB**: Database management for storing user information, property listings, inquiries, and booking data.
* **Express.js**: Backend framework that powers the API and server-side logic.
* **React.js**: Frontend framework to create a dynamic, responsive user interface.
* **Node.js**: Runtime environment for executing backend JavaScript and handling server requests.

**Additional Technologies**

* **Bootstrap, Material UI, Ant Design**: CSS frameworks for responsive, aesthetic UI components.
* **JSON Web Token (JWT)**: For secure user authentication and session management.

**4. System Requirements**

**Hardware**:

* Windows 8 or higher machine with a stable internet connection (30 Mbps recommended).

**Software**:

* Node.js (latest version)
* MongoDB Community Server
* Two web browsers (for testing purposes)

**5. Features**

**Key Features**

1. **User Registration & Authentication**:
   * Renter and Owner user roles are supported with unique functionalities and permissions.
   * Admin approval is required for owner accounts.
2. **Property Listings**:
   * Owners can add, edit, and delete property listings.
   * Each listing includes details like location, rent, property type, and availability status.
3. **Search and Filter**:
   * Renters can filter property listings by criteria like location, rent range, and type.
4. **Inquiry & Booking System**:
   * Renters can inquire about properties, and owners can accept or decline booking requests.
5. **Admin Dashboard**:
   * Admins review and approve owner registrations, manage reported issues, and monitor user activities.
6. **Real-Time Status Updates**:
   * Users receive instant updates on property availability, inquiry status, and booking confirmations.

**6. Project Architecture**

The application is divided into two main parts, client-side (frontend) and server-side (backend):

1. **Frontend (Client-Side)**:
   * Built with React.js for dynamic rendering.
   * Styled with Bootstrap, Material UI, and Ant Design for a clean and responsive design.
   * Communicates with the backend API for user interactions and data retrieval.
2. **Backend (Server-Side)**:
   * Uses Express.js to handle API requests and define routes for data manipulation.
   * Connected to MongoDB for database operations.
   * JSON Web Tokens (JWT) for secure authentication and session handling.
3. **Database**:
   * MongoDB for storing user data, property details, inquiries, and booking information.

**7. Installation and Setup**

**Prerequisites**

1. Install **Node.js** and **npm**.
2. Install **MongoDB Community Server**.

**Step-by-Step Setup**

1. **Clone the Repository**

<https://github.com/AkshayaGopalakrishnan/HOUSE-RENT-APP-USING-MERN.git>

1. **Backend Setup**:

Navigate to the backend folder and install dependencies:

* cd house-rent-app/code/backend
* npm install
* Create an .env file with MongoDB connection and JWT key.
* Modify the MongoDB connection string in connect.js in config folder
* Start the backend server:  
         npm start

1. **Frontend Setup**:

Navigate to the frontend folder and install dependencies:

* cd house-rent-app/code/frontend
* npm install
* Start the frontend server:  
        npm start

1. **Access the App**:
   * Frontend: http://localhost:3000
   * Backend: http://localhost:8000

**8. Workflow and Usage**

**User Roles and Functionalities**

1. **Renter**:
   * Registers or logs in, browses properties, applies filters, and inquires about specific listings.
2. **Owner**:
   * Registers, awaits admin approval, then adds and manages property listings.
3. **Admin**:
   * Reviews new owner registrations, monitors user activities, and enforces platform rules.

**9. Folder Structure**

house-rent-app/

├── frontend/

│   ├── src/                # React components and main logic

│   └── public/

├── backend/

│   ├── config/             # Database and environment configuration

│   ├── controllers/        # API route controllers

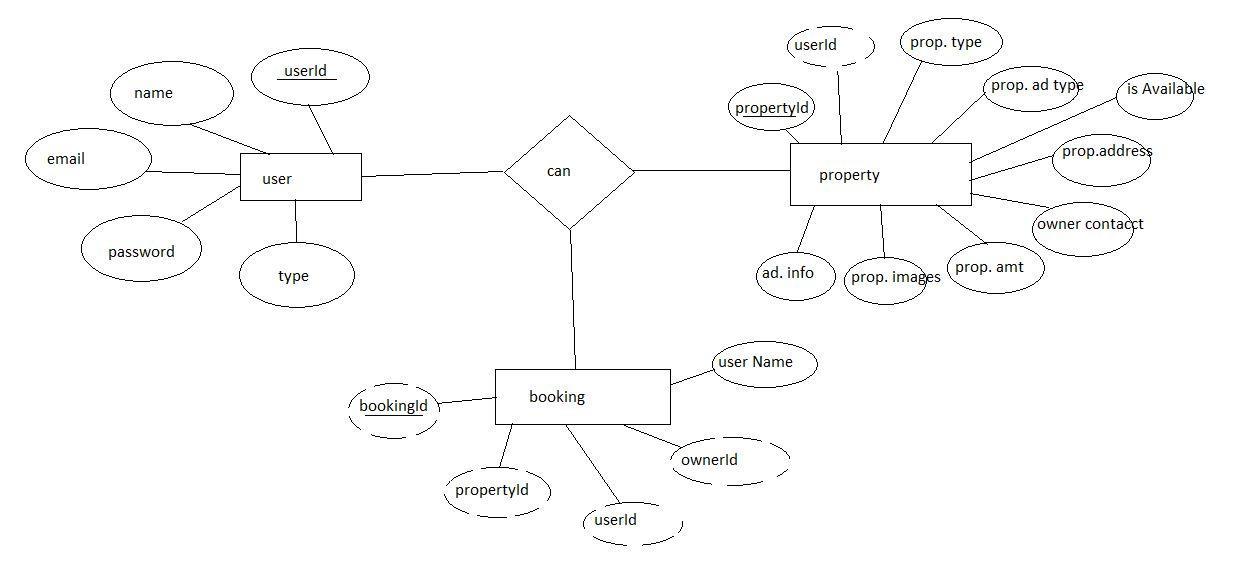
│   ├── models/             # MongoDB models

│   ├── routes/             # Route definitions

│   └── middleware/         # Authentication and validation

└── README.md

**10. ER Diagram**



**11. Challenges Faced**

1. **User Role Management**: Ensuring that each user role (Renter, Owner, Admin) has restricted access to only relevant features and data.
2. **Database Design**: Structuring MongoDB collections to support quick retrieval of properties and smooth handling of booking inquiries.
3. **Frontend Design**: Creating a responsive UI that maintains usability across devices and clearly differentiates between roles.

**12. Future Enhancements**

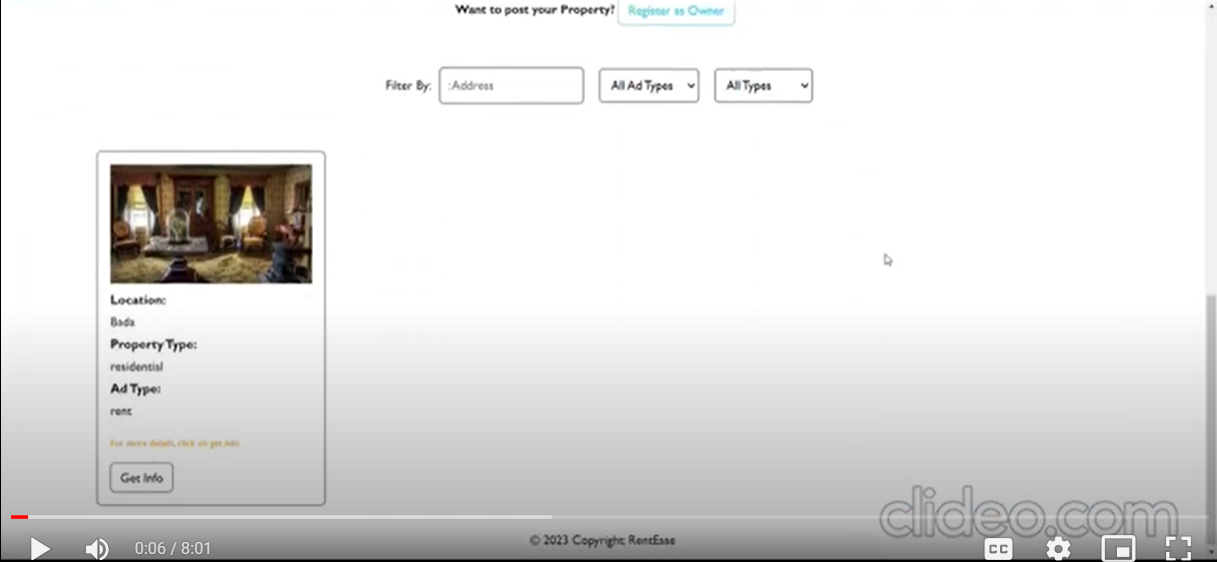
1. **Mobile Compatibility**: Developing a dedicated mobile app to increase accessibility.
2. **Advanced Filtering**: Adding more search criteria like amenities, neighbourhood ratings, and proximity to landmarks.
3. **Payment Integration**: Enabling secure online payments for booking confirmations.
4. **In-App Chat**: Allowing renters and owners to communicate directly within the platform.

**13. Project Implementation & Execution**

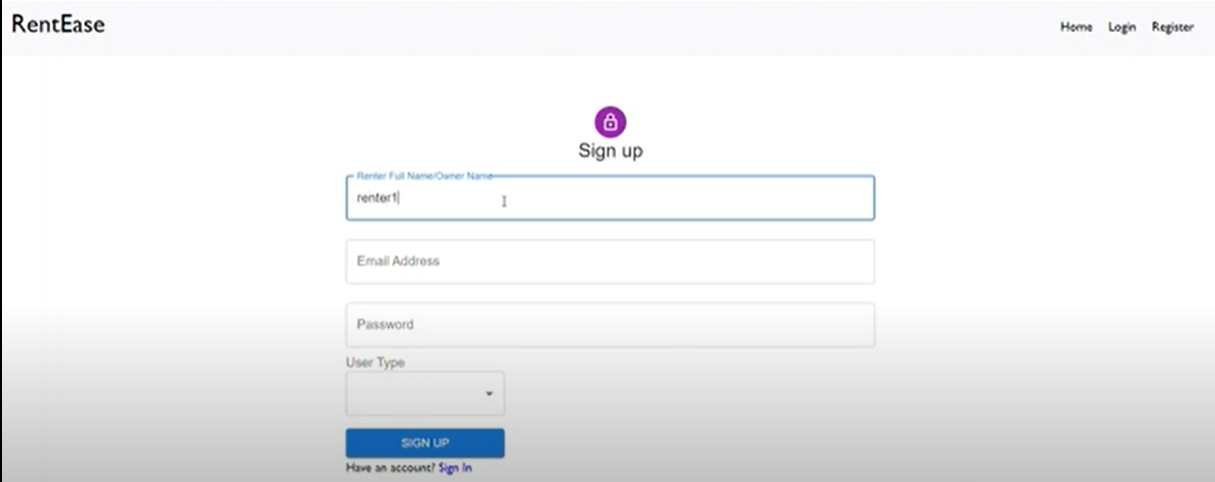
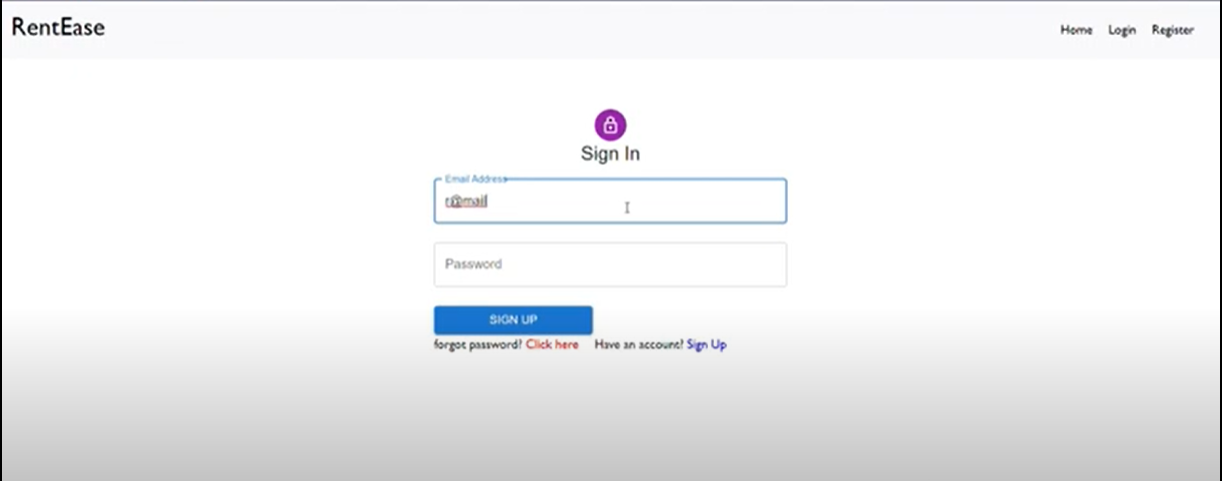
On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one’s provided below.

* **Landing page:**

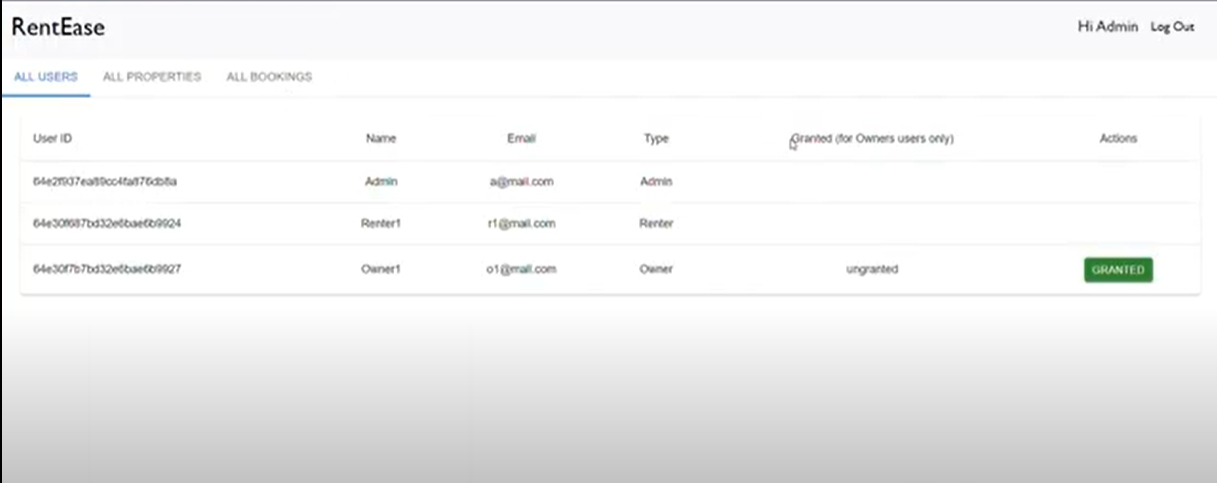
****

****

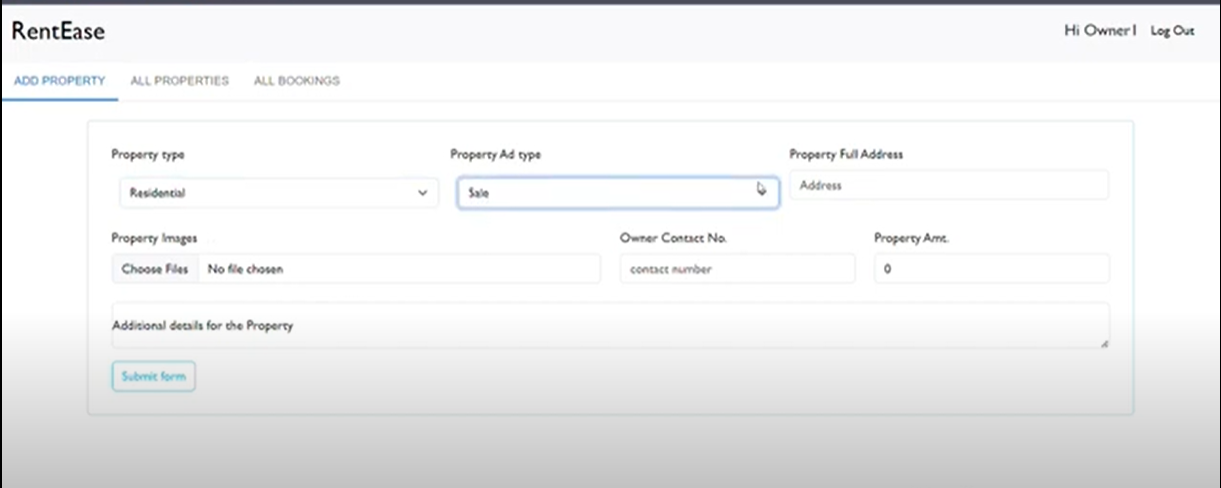
* **Login and register page:**

****

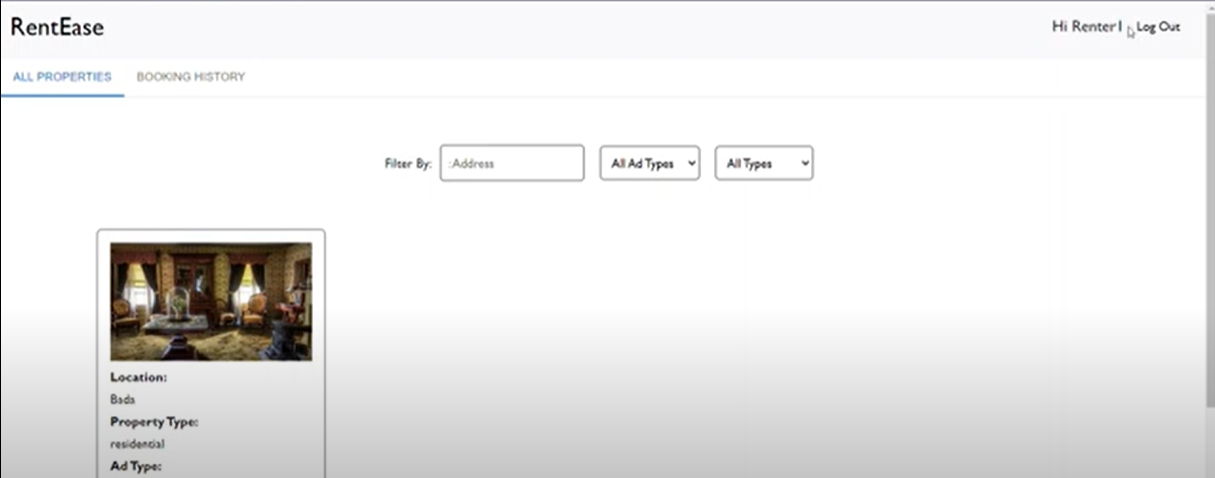
* **Admin Panel:**

****

* **Owner Panel:**

****

* **Tenant panel:**

****

**14. Conclusion**

The House Rent App is a robust solution designed to improve the rental experience for both property owners and renters. By focusing on user roles, secure authentication, and a comprehensive admin dashboard, this project successfully meets its primary objectives. Future updates will further enhance the app's functionality and usability, making it an invaluable tool in the property rental market.