**HOUSE RENT APP USING MERN**

1. **Introduction**

PROJECT TITLE : HOUSE RENT APP USING MERN

A house rent app is typically a mobile or web application designed to help users find rental properties, apartments, or houses for rent. These apps often offer features to make the process of searching for and renting a property more convenient and efficient.

This project was built by a team of dedicated professionals, each specializing in different areas.

The Project Manager [Mohanapriya C], leads the project with a focus on end-to-end development and integration.

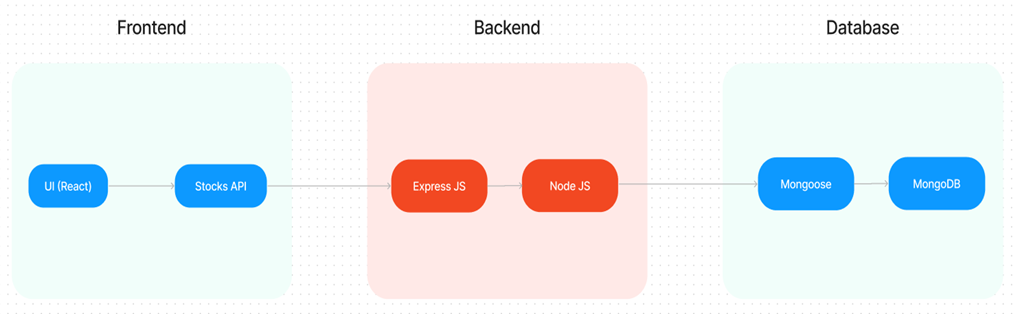
The Backend Developer, [Mythily R], focuses on server-side operations, ensuring smooth data management and security.

The Frontend Developer, [Sasirekha H], implements and refines the user interface, while the UI/UX Designer,

**2. Project Overview**

* **Purpose**:  
  The House Rent App is a web application designed to help users find rental properties, whether they are looking to rent or list a property for rent. Landlords can create listings for their properties, and tenants can browse available listings based on location, price, and amenities. The app offers a user-friendly interface with a secure authentication system for both landlords and tenants.
* **Features**:
  + **User Authentication**: Allow users to register, log in, and manage their accounts.
  + **Property Listings**: Landlords can add, edit, and delete rental listings with details such as price, location, description, and photos.
  + **Search & Filters**: Tenants can search for available properties and apply filters (location, price range, number of bedrooms, etc.).
  + **Property Details**: Each property has a detailed page showcasing images, descriptions, and contact information.
  + **Admin Panel**: Admin can view and manage all properties, users, and report inappropriate listings.
  + **Favorites**: Tenants can save their favorite properties for quick access later.
  + **Responsive Design**: The app is designed to be responsive and works across desktop and mobile devices.

1. **Architecture**

* 

Frontend (React):  
The frontend of the application is built using React.js. It is a single-page application (SPA) that uses React Router for navigation between pages (e.g., Home, Property Listings, Property Details, Profile). State is managed using React’s Context API, and React hooks are used for component lifecycle management. Styling is handled using CSS modules and/or styled-components for a modular, scalable approach.

* Backend (Node.js & Express.js):  
  The backend is built with Node.js and Express.js. Express handles API requests and serves as a middle layer between the frontend and the database. The backend is responsible for user authentication, CRUD operations for properties, and handling requests from the frontend.
* Database (MongoDB):  
  MongoDB is used as the database to store user information, property listings, and other necessary data. Mongoose is used for modeling and interacting with the database. The database consists of two primary collections:
  + Users: Stores user information (name, email, password, role, etc.).
  + Properties: Stores rental property information (address, price, description, images, etc.).

1. **Setup Instructions**

**Prerequisites:**

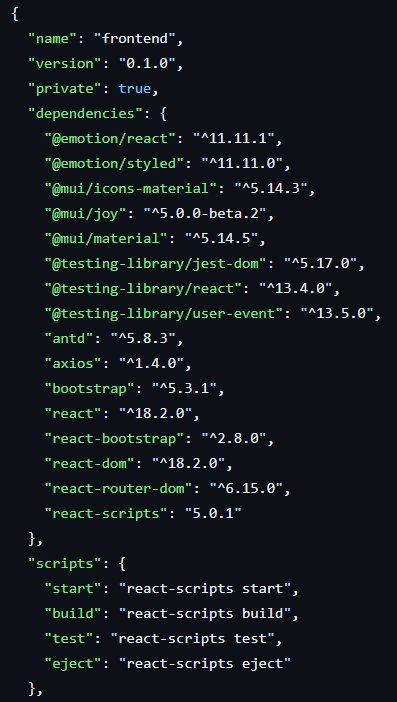
* Node.js (version 14.x or higher)
* MongoDB (local or cloud instance, e.g., MongoDB Atlas)
* npm or yarn for managing project dependencies
* A code editor (e.g., Visual Studio Code)

**Installation:**

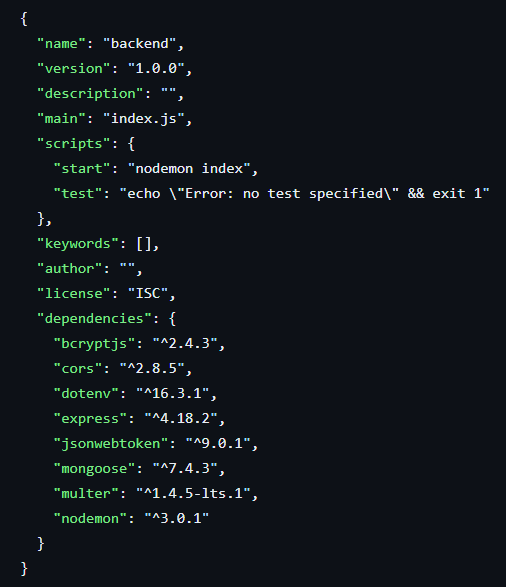
1. Clone the repository:  
   git clone https://github.com/your-repo/house-rent-app.git
2. Navigate to the project directory:  
   cd house-rent-app
3. Install frontend dependencies:  
   cd client && npm install
4. Install backend dependencies:  
   cd server && npm install
5. Set up environment variables:
   * Create a .env file in the server directory with the following:
     + MONGODB\_URI=<Your MongoDB connection string>
     + JWT\_SECRET=<Your JWT secret key>
     + PORT=5000
   * Create a .env file in the client directory with the following:
     + REACT\_APP\_API\_URL=http://localhost:5000

**5. Folder Structure**

**Client:**



**Server:**



The folder structure is organized to maintain a clean and modular architecture.

**Client:** This folder contains subfolders for components (for UI elements), pages assets (for images and styling), and services (for handling API requests). This organization keeps the React components modular and easier to update.

**Server**: This folder includes the routes (API endpoints), controllers (business logic), models (MongoDB schemas), middleware (for security functions like authentication), and configuration files. This modular design allows each part of the backend to handle specific functions, simplifying the development process.

**6. Running the Application**

To run the application locally:

* + - Start the frontend server by navigating to the client directory and executing npm start.
    - Start the backend server by navigating to the server directory and executing npm start.

**7. API Documentation**

* POST /api/auth/register  
  This endpoint allows new users (either tenants or landlords) to register for the app by providing their name, email, password, and role. The role can either be "tenant" or "landlord". Successful registration creates a new user in the database, while errors may include duplicate emails or missing fields.
* POST /api/auth/login  
  The login endpoint is used by registered users to authenticate and obtain a JWT token. This token is required for making authenticated requests. Users need to provide their email and password, and if the credentials match, a token is returned. If authentication fails, an error message is sent.
* GET /api/auth/me  
  This endpoint fetches the currently authenticated user's profile using the JWT token provided in the request headers. The response includes the user's basic details like ID, name, email, and role. Unauthorized access or expired tokens result in an error.

**8. Authentication**

Users authenticate via JWT (JSON Web Tokens). On successful login, a JWT token is issued and returned to the frontend. For protected routes (e.g., adding a property), the token is sent in the Authorization header. Token is stored in the frontend (local Storage or cookies), and it's sent with each subsequent request.

**9. User Interface**

The user interface is built with a focus on accessibility and ease of navigation. checkout pages provide a clear representation of the user journey from browsing to purchasing, showcasing a consistent and streamlined experience.

* The **Home Page** displaying property listings.
* The **Property Details Page** showcasing details and images of a listing.
* **Login/Signup** forms for user authentication.

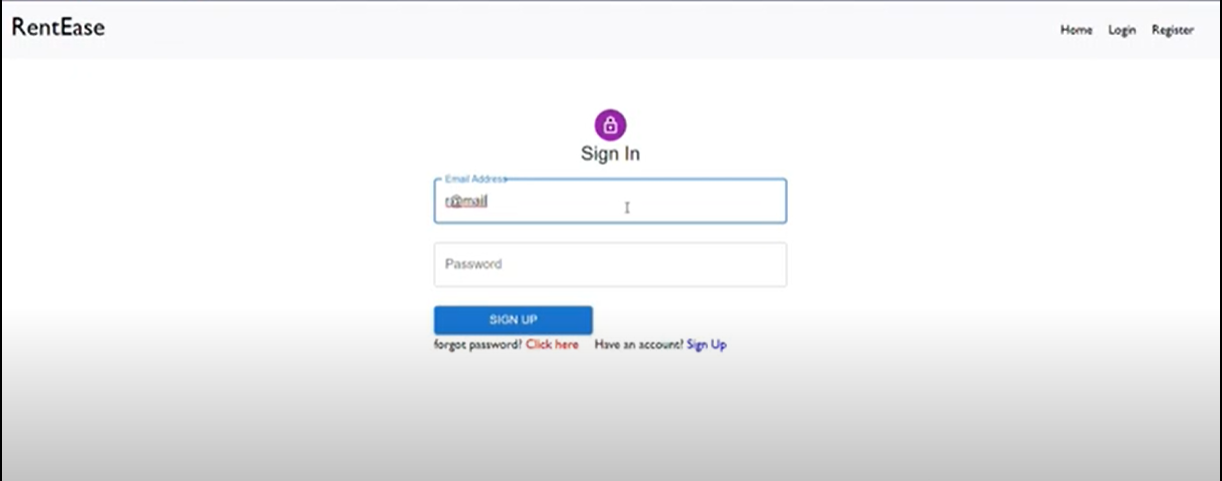
**10. Testing**

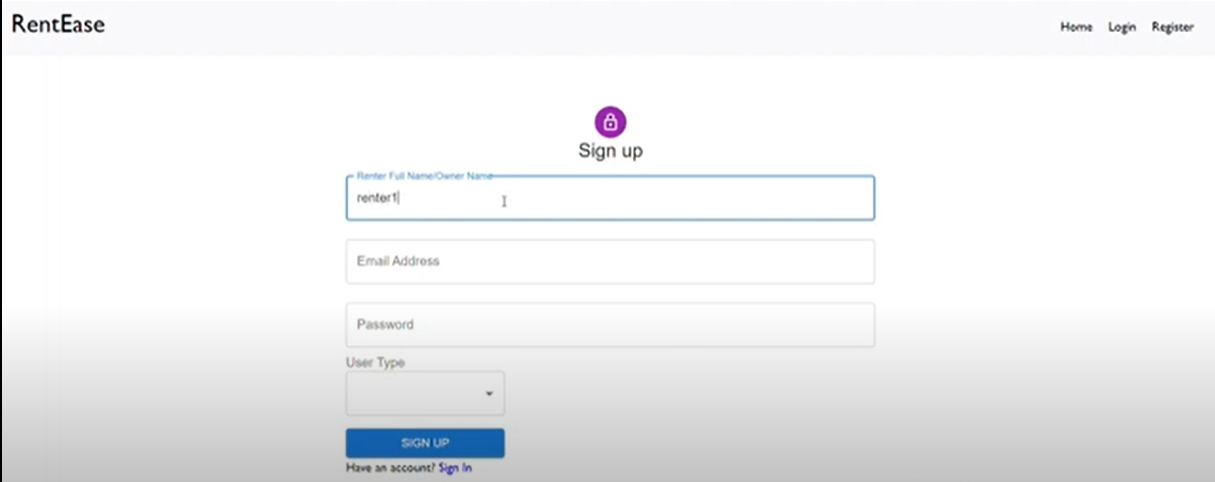
* + - Frontend: Jest and React Testing Library are used to test React components and user interactions.
    - Backend: Mocha and Chai are used to test the API routes (GET, POST requests) and controllers.
    - End-to-End: Cypress is used to perform end-to-end testing to simulate user flow.

**11. Screenshots**

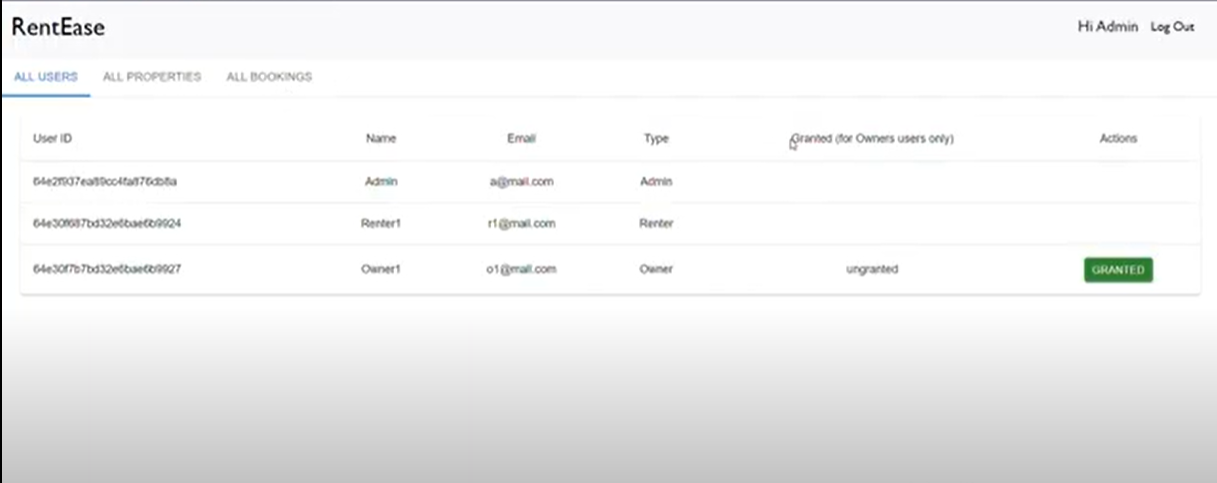


LOGIN AND REGISTER PAGE:

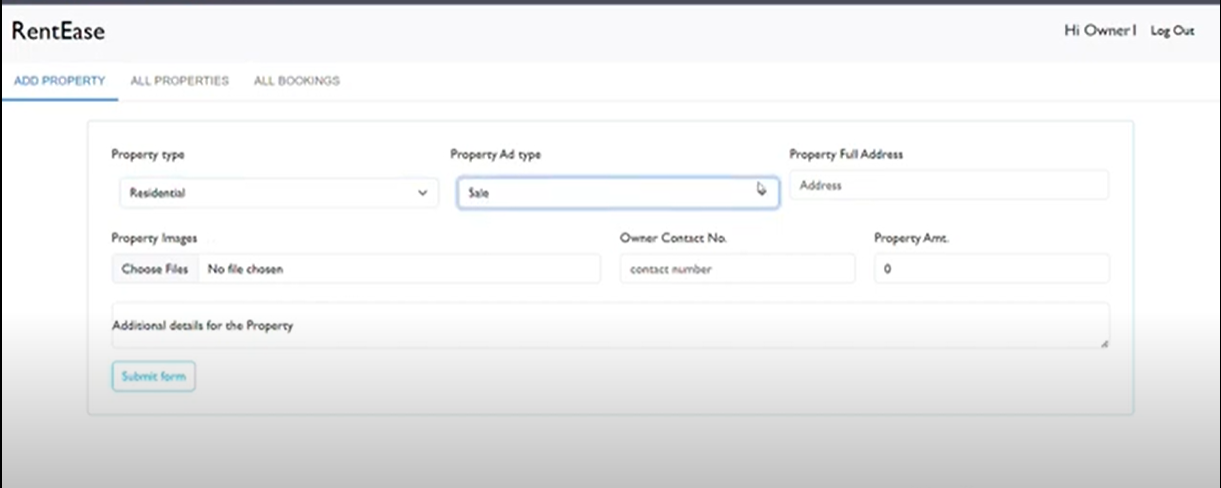




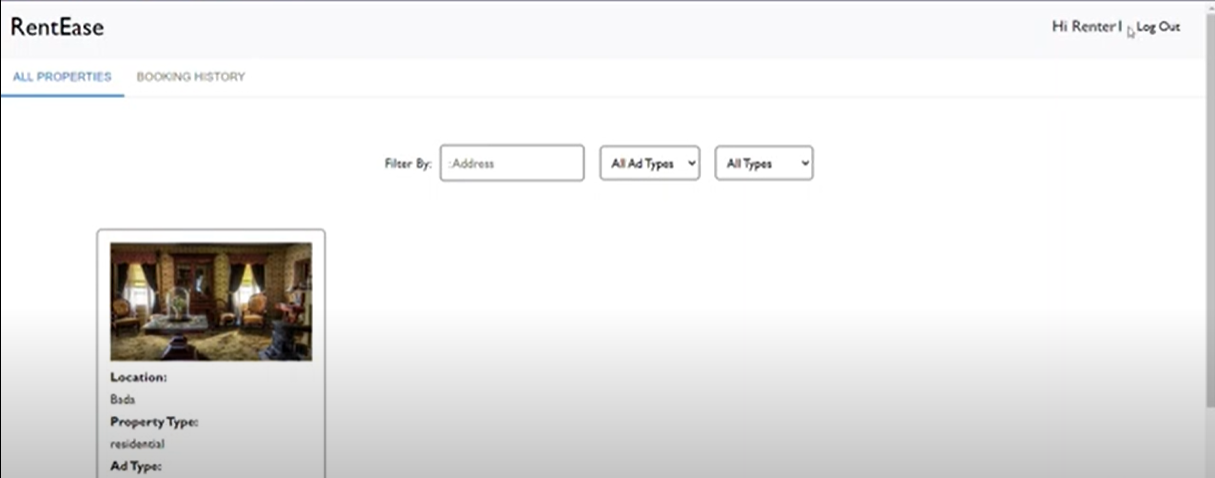
ADMIN PANEL

****

OWNER PANEL



TENANT PANEL



**12. Known Issues**

The app currently has a few known issues that are being addressed:

* Property Image Upload: There is a known issue with image uploads where large images sometimes fail to upload.
* Search Filters: Occasionally, search filters may not update the listing results immediately after applying multiple filters.

**13. Future Enhancements**

To continuously improve the app, several future enhancements are planned, including:

Implement an Admin Dashboard for managing users, properties, and reports.

* Add Payment Integration to allow tenants to pay rent directly through the app.
* Add Real-time Chat between tenants and landlords for inquiries and property discussions.
* Implement a Rating & Review System for tenants to rate properties after renting

This detailed report captures the project’s key features, technical setup, and functionality, providing a comprehensive reference for users and developers alike to understand, set up, and contribute to the grocery web app.

.