

NAAN MUDHALVAN PROJECT

MERN stack powered by MongoDB



PROJECT TITLE

House Rent Application using MERN Stack

Submitted by the team members of Final Year CSE 'B'

SUMAIYA Q	311421104099
SANJANA V	311421104081
YUVA SHREE B	3114211040113
VINODHA S	3114211040110

ABSTRACT

The House Rent Application (HRA) is an innovative solution built on the MERN stack (MongoDB, Express, React, Node.js), designed to simplify and streamline the process of renting properties for tenants and landlords. The platform ensures an accessible, efficient, and user-friendly experience for managing property rentals.

HRA features a robust client-server architecture supporting real-time data handling for functionalities such as user authentication, property listings, search and filter options, and secure payment processing for rental transactions. Rolebased access allows landlords to list and manage properties, tenants to search and apply for rentals, and administrators to oversee platform operations effectively.

The application is accessible across devices, offering an engaging user experience that empowers tenants to find suitable homes, interact with landlords, and complete the renting process seamlessly. Landlords can efficiently manage property listings, track rental inquiries, and communicate with prospective tenants.

This documentation outlines the requirements, architecture, features, and implementation strategy for developing HRA using MERN. It serves as a foundational guide for creating a modern and effective property rental ecosystem. Keywords: Property Rental, Web Application, MERN Stack, House Rent App

TABLE OF CONTENT

Team Information.....	5
Introduction.....	5
Project Overview	6
Purpose	6
Key Features	6
Technical Architecture	7
Frontend.....	7
Backend	8
Database.....	8
Setup Instructions.....	8
Prerequisites.....	8
Installation Steps.....	9
Application Structure	9
Frontend Structure.....	9
Backend Structure.....	10
Running the Application	10
Development Mode.....	10
API Documentation	11
Authentication.....	11
1. Register User.....	11
o Route: POST /api/auth/register	11
o Description: Allows new users to register by providing name, email, password, role, and phone number.....	11
2. Login User	11
o Route: POST /api/auth/login oDescription: Authenticates users with email and password, returning a JWT token on success.....	11
Properties	11
List All Properties	11
• Route: GET /api/properties.....	11
• Description: Retrieves all available properties for renters	11
Add Property	11
• Route: POST /api/addproperties	11

• Description: Allows property owners to add new properties, including details such as address, price, owner contact, and other information	11
Get Properties by Owner	11
• Route: GET /api/properties/:ownerId.....	11
• Description: Fetches all properties added by a specific owner	11
Update Property	11
• Route: PUT /api/properties/:id	11
• Description: Allows owners to update property details	11
Delete Property	11
• Route: DELETE /api/properties/:id	11
• Description: Enables owners to remove a property from the platform	11
Bookings	12
Admin	12
Middleware	13
Security & Authentication	13
User Interface Overview	13
Landing Page (Hero Section).....	13
Authentication	13
Dashboards	13
Known Issues	14
Future Enhancements	14
Conclusion.....	21

House Rental Management System Using MERN

Team Information

- **Project Title:** *House Rental Management System*
- **Team ID:** NM2024TMID00530
- **Team Members:**

Sumaiya Q – Full stack developer [*Team Lead*]

Yuva shree B– Project Management and Documentation

Vinotha S – Background research and Design

Sanjana V – Frontend developer

Introduction

The **House Rental Management System** is a comprehensive, web-based platform developed using the **MERN** stack (MongoDB, Express.js, React, Node.js), designed to streamline and centralize the rental process. It offers a one-stop solution for **property owners, renters, and administrators**, each benefiting from customized interfaces and capabilities that enhance usability and efficiency. Property owners can easily list rental properties, complete with descriptions, photos, and rental details, creating a seamless experience for showcasing and managing properties. Through the **Owner Dashboard**, owners can review incoming booking requests and approve or reject them, maintaining control over property bookings without unnecessary complications. Renters can browse available properties, filtering for specific requirements like location, price range, and property type. By providing an intuitive booking request process, the system ensures that renters can submit booking requests directly to owners and stay informed of booking statuses. This simplifies the rental journey and minimizes the back-and-forth typically involved in property inquiries. Administrators have complete oversight of the system. They can verify owners, manage listings, and supervise user activities across the platform. The **Admin Dashboard** gives centralized access to all properties,

owners, and renters, allowing administrators to approve or disapprove listings, ensure platform security, and address user concerns.

Security is a central focus of the platform, which uses **JWT-based** authentication to safeguard user data and ensure secure access control. Each role—whether renter, owner, or admin—has access to specific features and views based on their permissions, enhancing data integrity and providing a more tailored user experience. The backend's secure **RESTful APIs** support this role-based functionality, connecting the frontend and database while maintaining streamlined data processing. Additionally, the platform is built with scalability in mind, supporting features that enable seamless user management, real-time booking updates, and notifications, making it a reliable solution for both small and large property rental operations. This architecture not only provides an efficient rental experience but also establishes a solid foundation for potential future expansion, such as integrating advanced search filters, in-app messaging, and payment processing/

Project Overview

Purpose

The House Rental Management System connects property owners and renters on a single platform, optimizing the rental process by automating property listings, bookings, and management. It ensures transparency, security, and efficiency by streamlining tasks that were previously manual, improving user experience, and enhancing operational workflows.

Key Features

1. Role-Based Dashboards:

- **Owners:** Can list properties, manage their bookings, and view all requests. They control the availability and booking status of their properties.
- **Renters:** Can explore available properties, submit booking requests, and track their booking statuses in real-time.
- **Admins:** Have complete oversight over platform operations, verify ownership, manage user access, and ensure the smooth functioning of the system.

2. Comprehensive Property Management:

- **Property Creation and Management:** Owners can create, update, and delete property listings, including uploading images and detailed property specifications (location, price, type, etc.).
- **Real-Time Availability:** Property status and availability are updated instantly, ensuring accurate and up-to-date information for both owners and renters.

3. User Management & Security:

- **Role-Specific Access:** Users are assigned different access levels based on their roles (owner, renter, admin). This ensures that users only see the features and data relevant to them.
- **Owner Verification:** Admins can approve or reject owners based on verification, ensuring the legitimacy of property listings.
- **Secure Sessions:** All users authenticate through secure login processes, with passwords hashed for protection and token-based authentication ensuring session safety.

4. Booking Process Automation:

- **Easy Booking Requests:** Renters can easily submit booking requests for properties, with details about their contact information and rental dates.
- **Status Tracking:** Once a booking request is submitted, renters can track its status. Owners can approve or deny requests, and the system automatically updates the status for renters, ensuring clear communication between both parties.

Technical Architecture

Frontend

The **React**-based frontend is optimized for responsive and intuitive user interaction, and is divided into primary components:

- **Home Page (Hero Section):** Displays featured properties, a "Get Started" button, and application overview.

- **Registration and Login Forms:** Validation and submission forms for new and returning users.
- **Dashboard Pages:** Role-based interfaces for owners, renters, and administrators.
- **Property & Booking Management Tools:** Allows easy navigation and management of properties and bookings.

Backend

The **Node.js** and **Express.js** backend provides core functionality through secure RESTful APIs:

- **User Authentication:** Supports role-based access control for different dashboards.
- **Property & Booking Management:** RESTful endpoints for CRUD operations on properties and bookings.
- **Admin Capabilities:** Enhanced admin functionalities, including user verification and property oversight.
- **Secure API:** Protected endpoints using **JWT** for authenticated and authorized access.

Database

The **MongoDB** database stores all application data, organized with **Mongoose** models:

- **Users:** Stores profiles for owners, renters, and admins with relevant fields.
- **Properties:** Contains listings with property details like cost, address, owner contact.
- **Bookings:** Tracks renter requests, timestamps, and status updates.

Setup Instructions

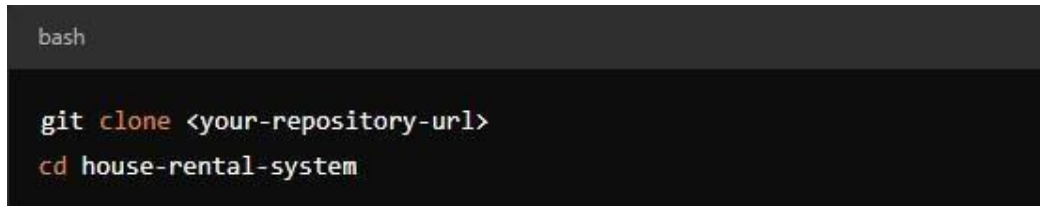
Prerequisites

- **Node.js** (v14 or higher)
- **MongoDB** (v4.4 or higher)
- **npm** (v6.14 or higher)

Installation Steps

Follow these steps to perform installation:

1. **Clone the Repository:**

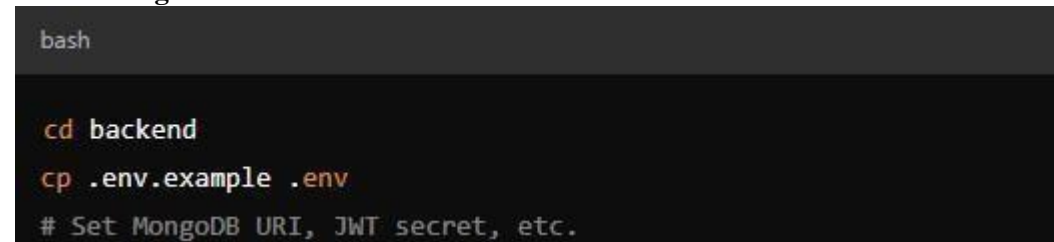


```
bash

git clone <your-repository-url>
cd house-rental-system
```

Fig 1- commands to clone repository

2. **Environment Configuration:**



```
bash

cd backend
cp .env.example .env
# Set MongoDB URI, JWT secret, etc.
```

Fig 2- Command to create environment variables

3. **Install Dependencies:**

- o **Backend:** cd backend && npm install

- o **Frontend:** cd ../frontend && npm install

Application Structure

Frontend Structure

- **public/:** Contains static assets.
- **src/:**
 - **components/:** All component files such as HeroSection.jsx, OwnerDashboard.jsx, BookingForm.jsx, RenterDashboard.jsx, AdminDashboard.jsx.
 - **hooks/:** Contains custom hooks like useOwnerDashboard.js.
 - **styles/:** Contains CSS styles for various components.
 - **utils/:** Utility functions for frontend.
 - **App.js:** Main application component.
 - **index.js:** Application entry point.

Backend Structure

- **config/**: Configuration files, including connect.js for database setup.
- **controllers/**: Contains all logic handling for requests, including userController.js and propertyController.js.
- **middleware/**: Includes authMiddleware.js for JWT validation.
- **routes/**: API route definitions (e.g., userRoute.js, propertyRoute.js).
- **schemas/**: Mongoose schemas for userModel.js, propertyModel.js, and bookingModel.js.
- **.env**: Contains environment variables.
- **index.js**: Main backend file.

Running the Application

Development Mode

1. Start Frontend:

```
cd frontend
npm start
# Accessible at http://localhost:3000
```

Fig 3 -command to start the frontend

2. Start Backend:

```
cd backend
npm run dev
# API available at http://localhost:5000
```

Fig 4 – Command to run the backend

API Documentation

Authentication

1. *Register User*

- **Route:** POST/api/auth/register
- **Description:** Allows new users to register by providing name, email, password, role, and phone number.

2. *Login User*

- **Route:** POST /api/auth/login
- **Description:** Authenticates users with email and password, returning a JWT token on success.

Properties

1. List All Properties

- **Route:** GET /api/properties
- **Description:** Retrieves all available properties for renters.

2. Add Property

- **Route:** POST /api/addproperties
- **Description:** Allows property owners to add new properties, including details such as address, price, owner **contact, and other information.**

3. Get Properties by Owner

- **Route:** GET /api/properties/:ownerId
- **Description:** Fetches all properties added by a specific owner.

4. Update Property

- **Route:** PUT /api/properties/:id
- **Description:** Allows owners to update property details.

5. Delete Property

- **Route:** DELETE /api/properties/:id
- **Description:** Enables owners to remove a property from the platform.

Bookings

1. Create Booking

- **Route:** POST /api/renter/bookings
- **Description:** Allows renters to book a specific property by providing required details.

2. View Renter Booking History

- **Route:** GET /api/renter/booking-history
- **Description:** Fetches booking history for the logged-in renter.

3. View All Bookings for Owner's Properties

- **Route:** GET /api/bookings/:ownerId
- **Description:** Fetches all bookings for properties owned by a specific owner.

4. Update Booking Status

- **Route:** PATCH /api/bookings/:bookingid
- **Description:** Allows property owners to approve or reject a booking request.

5. View All Bookings for a Renter with Property Details

- **Route:** GET /api/renter/bookings
- **Description:** Fetches all bookings for a renter, populated with property details.

Admin

1. View All Users

- **Route:** GET /api/admin/all-renters
- **Description:** Fetches a list of all users on the platform.

2. View All Owners

- **Route:** GET /api/admin/all-owners
- **Description:** Approves or rejects an owner's account after verification.

3. List All Properties

- **Route:** GET /api/admin/all-properties
- **Description:** Provides a complete list of properties for administrative oversight.

4. Perform Owner Action

- **Route:** POST /api/admin/owner-action
- **Description:** Approves or rejects an owner's account after verification by admin.

Middleware

- **Authentication Middleware:** Verifies user tokens before accessing protected routes.
- **Error Handling :** Errors are logged using a logger for better debugging and are sent to the client with appropriate status codes.

Security & Authentication

- **JWT Authentication:** Secure access to restricted areas and APIs.
- **Password Hashing:** All passwords are securely hashed.
- **Role-Based Access Control:** Grants access based on user roles.
- **Other Security Features:** CORS configuration, rate limiting to prevent abuse.

User Interface Overview

Landing Page (Hero Section)

The **Hero Section** introduces the application with featured properties and a **Get Started** button for easy registration. Scrolling further reveals a brief **About** section and a list of properties available for rent.

Authentication

- **Register:** New users input their name, email, password, contact, and select a role (Owner or Renter).
- **Login:** Returning users log in with email and password.

Dashboards

1. Owner Dashboard:

- **Header:** Displays welcome message and logout.
- **Add Property:** Allows owners to Add new properties

- **All Properties:** Displays properties added by the owner.
- **All Bookings:** Displays booking requests with approve/reject options.

2. **Renter Dashboard:**

- **Header:** Shows welcome note and logout.
- **All Properties:** Shows available properties with booking options.
- **My Bookings:** Displays booking history and status updates.

3. **Admin Dashboard:**

- **Header:** Shows logout option.
- **All Properties:** Lists all properties across the platform.
- **All Owners:** Lists all owners with verification options.
- **All Renters:** Lists all renters with details.

Known Issues

- Image upload size limitations.
- Minor issues with mobile responsiveness.
- Potential for optimization in the search functionality.

Future Enhancements

- **Advanced Property Search:** Filter and map view for more precise searches.
- **In-App Messaging:** Facilitates communication between renters and owners.
- **Payment Integration:** Allows renters to make secure booking payments.
- **Mobile Application:** Native mobile apps for enhanced user experience.

Screenshots

Home Page

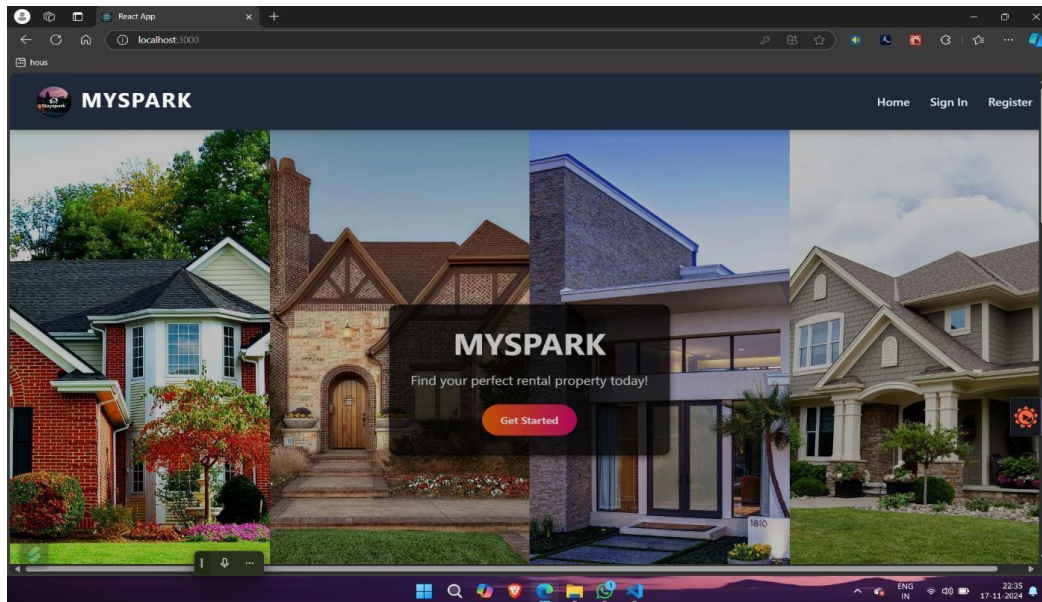


Fig -5 Hero section of our website

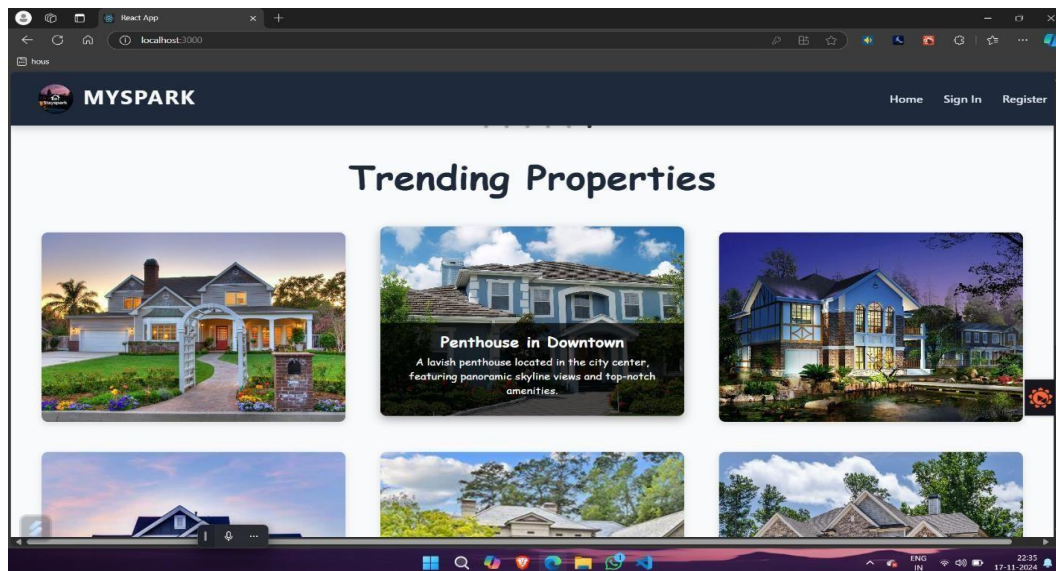


Fig-6 A section displaying the trending properties

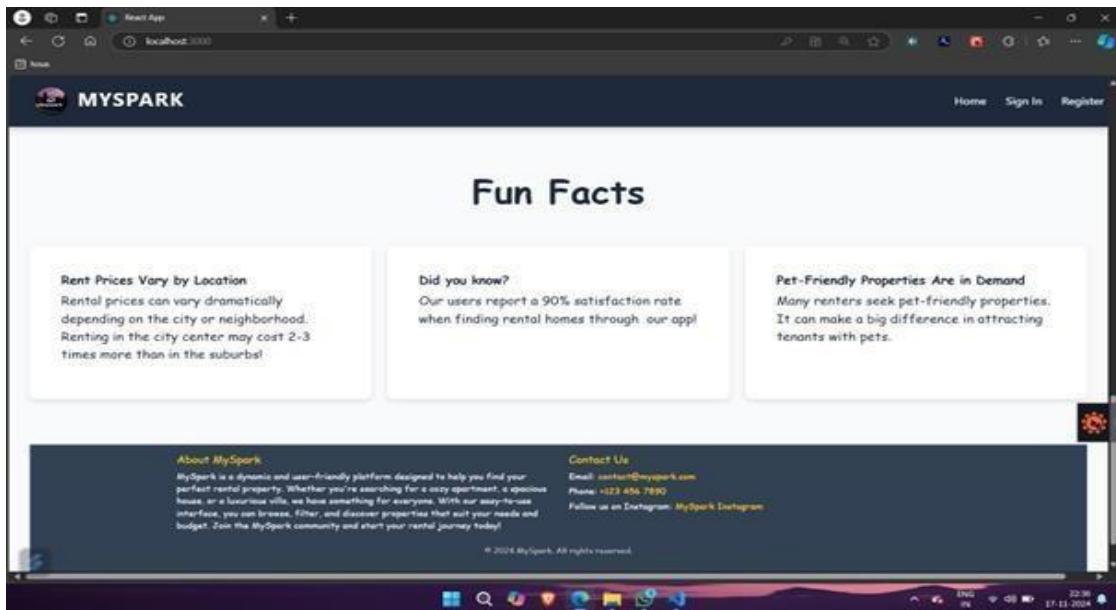


Fig-7 A section displaying fun facts and Footer

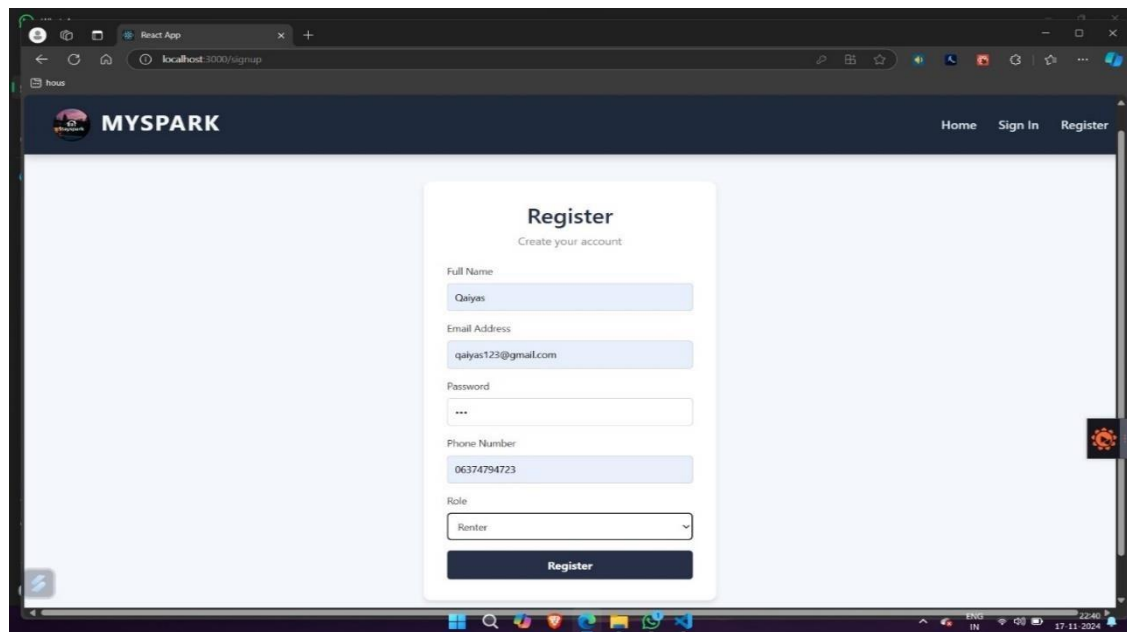


Fig-8 Register Page

Login page

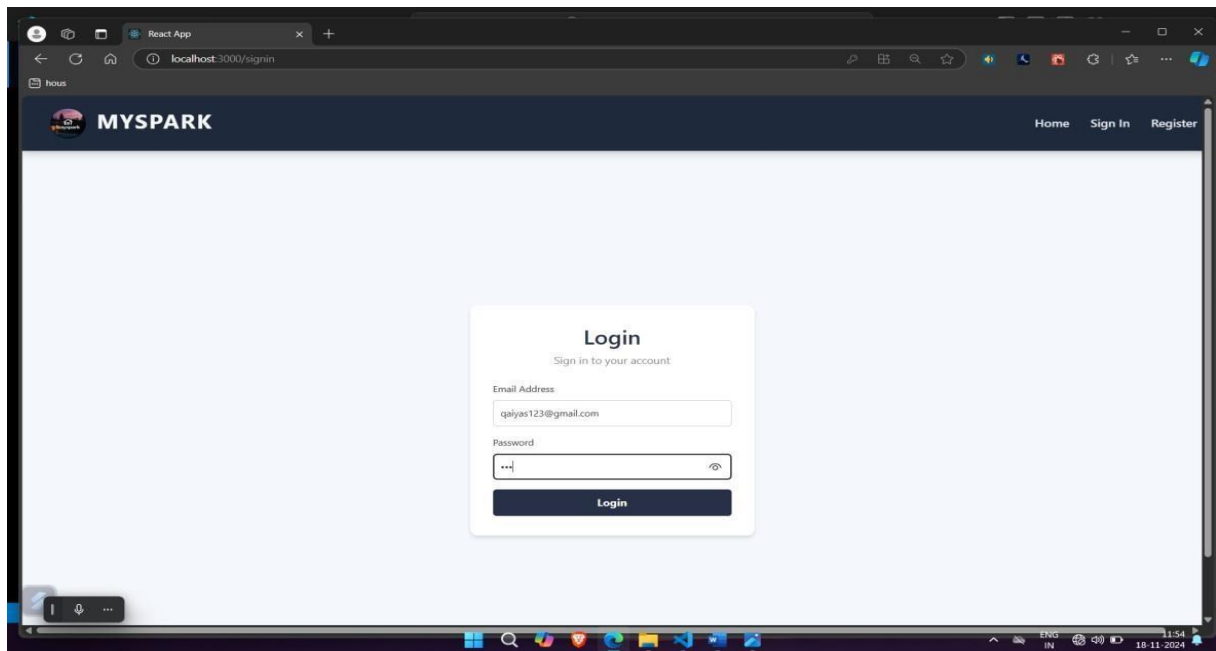


Fig-9 Login page

3.RENTER DASHBOARD

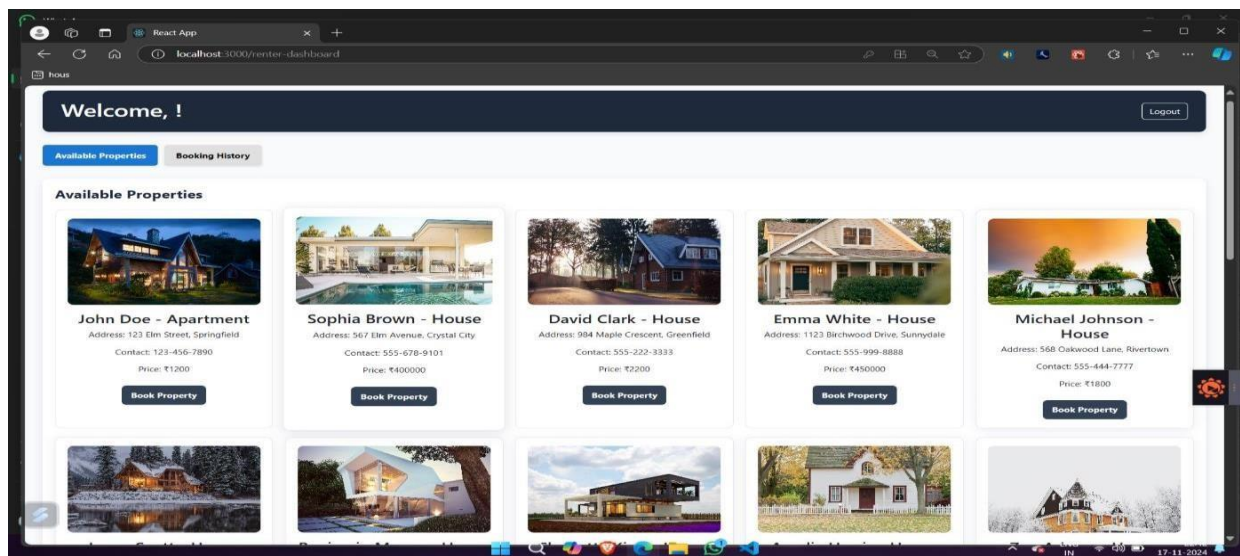


Fig-10 All properties being displayed in the renter dashboard

4.OWNER DASHBOARD

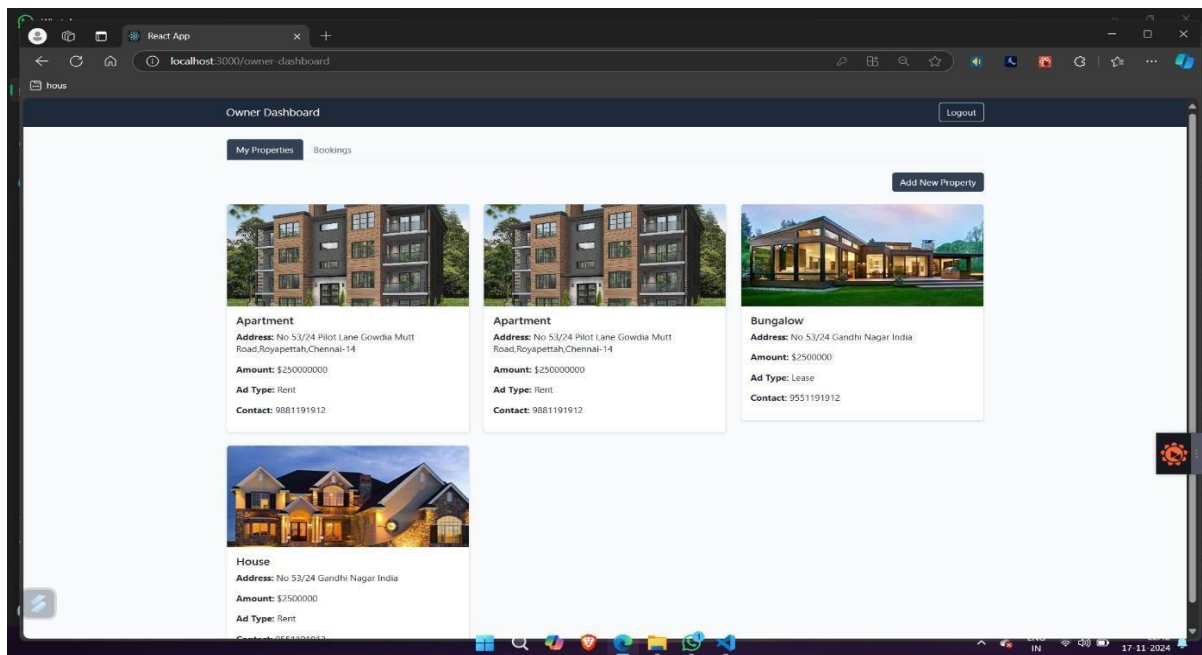


Fig-11 All Properties Tab in the owner Dashboard

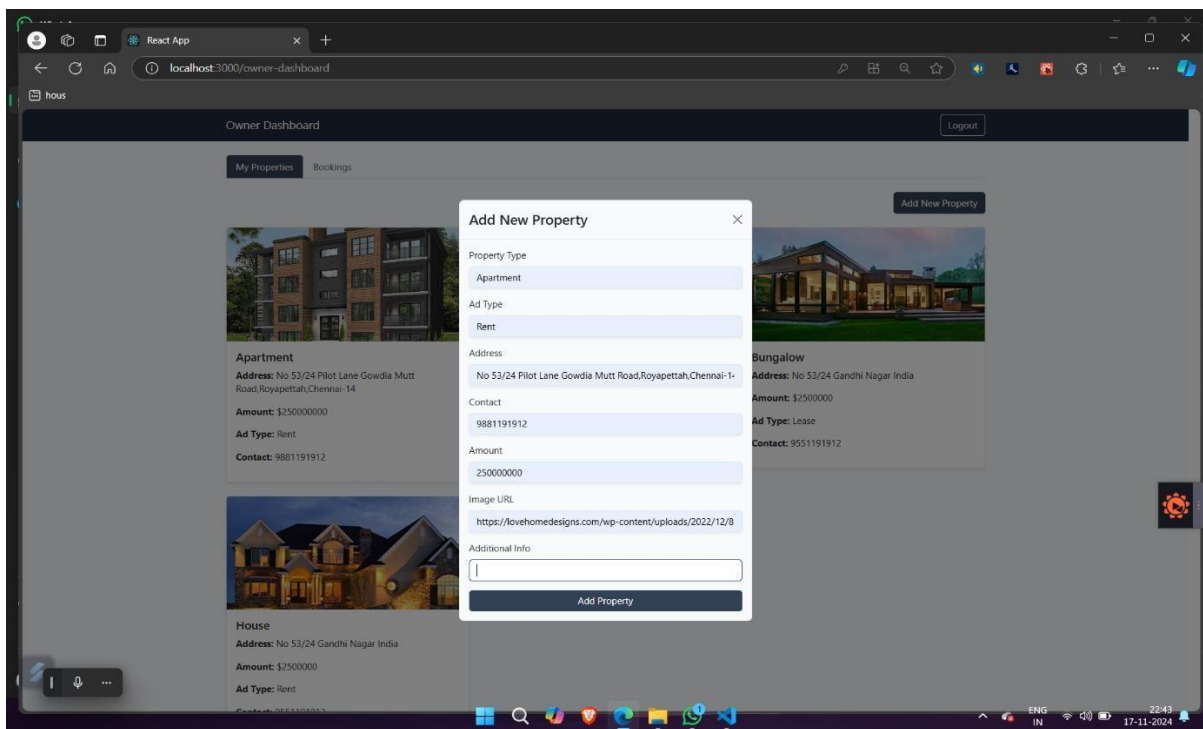


Fig – 12 Form to add a new property

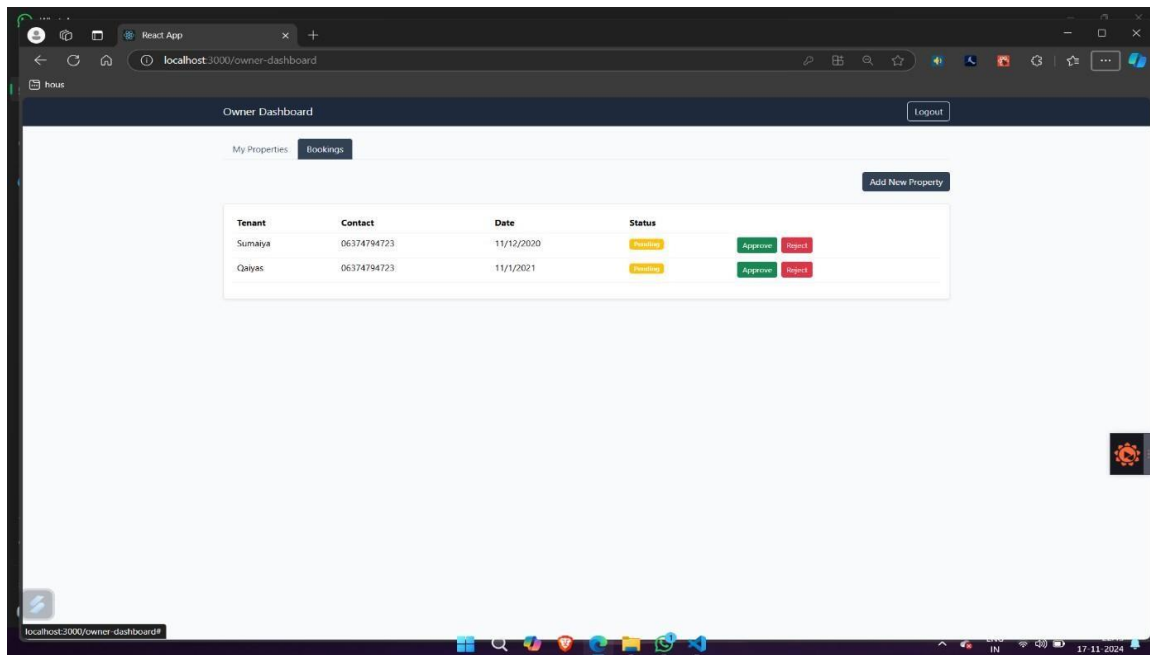


Fig 13-A tab displaying the booking requests received by the owner

4.ADMIN DASHBOARD

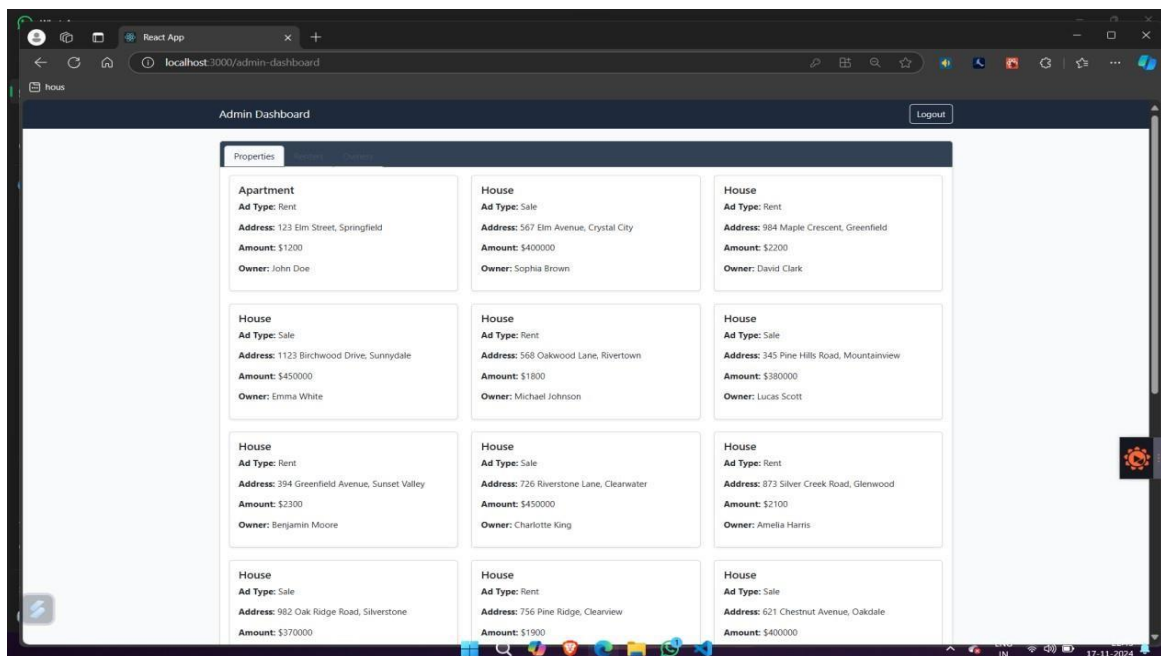


Fig 14 – A tab displaying all the properties

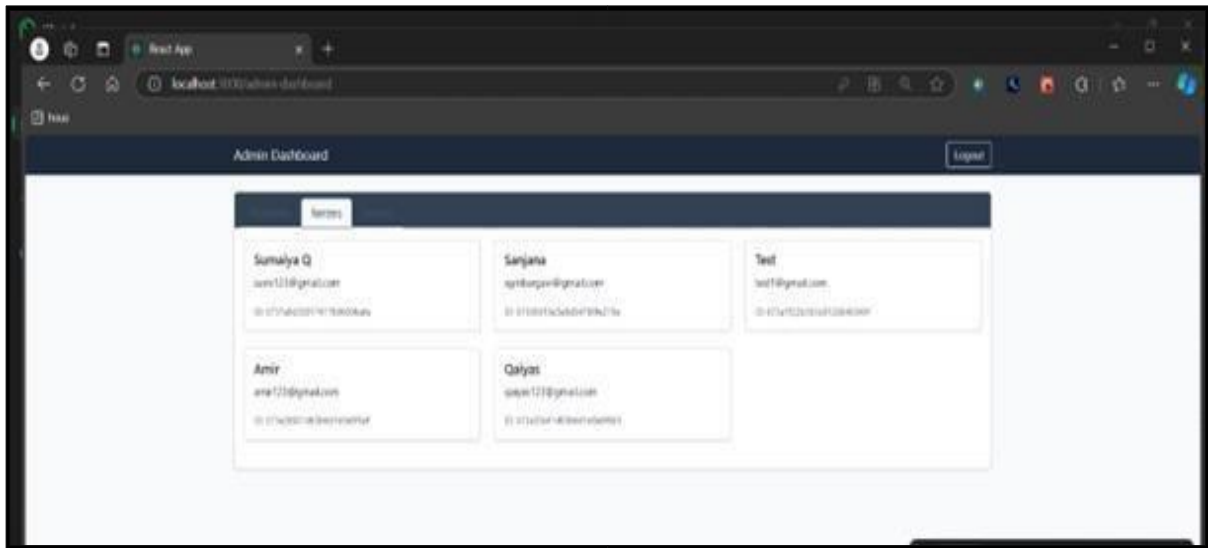


Fig 15 – A tab displaying all the renters

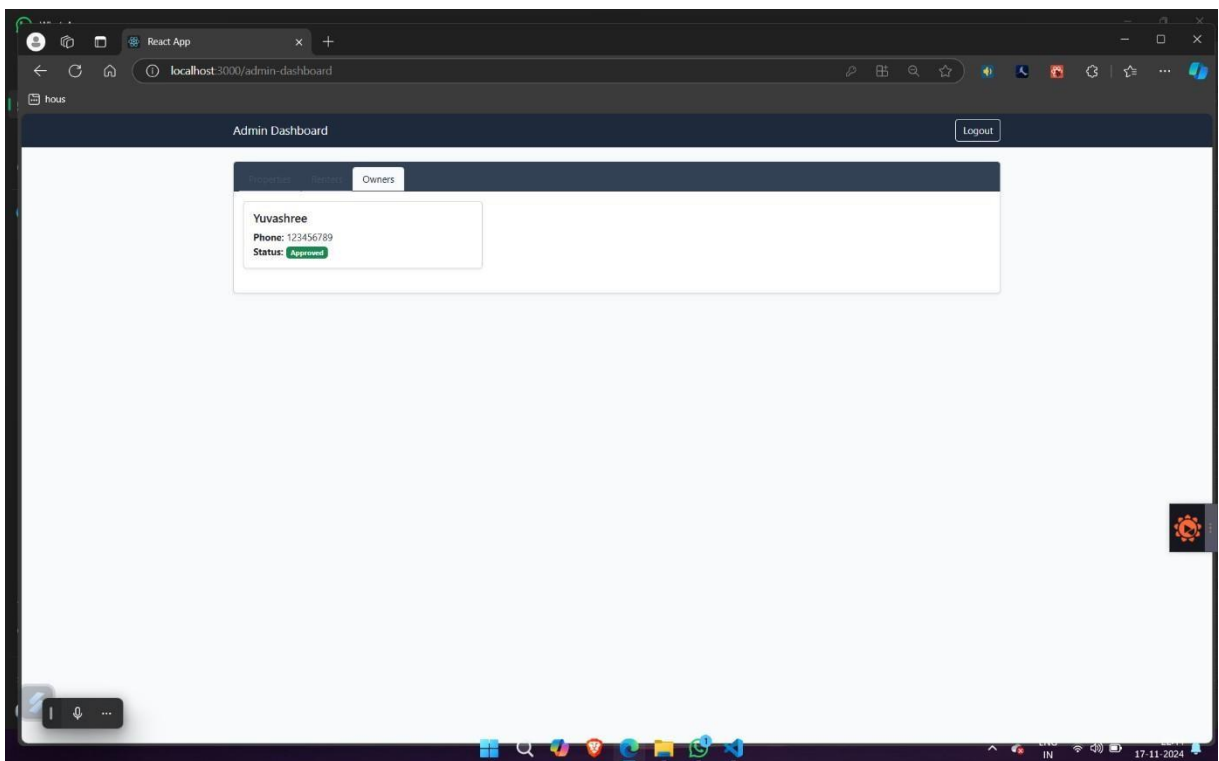


Fig 12 – A tab displaying all owners

Video Reference links:

- [House Rent App Video Link](#)

Conclusion :

This House Rental Management System exemplifies the versatility and efficiency of the MERN stack in building robust web applications. With a seamless blend of user-centric design, secure architecture, and modular coding practices, it offers a scalable solution for managing rental properties. The system's dynamic features ensure adaptability to future requirements, while its structured approach facilitates easy maintenance and enhancements. By prioritizing usability, security, and performance, the application sets a strong precedent for further innovation, demonstrating how modern technology can optimize everyday processes like property rentals.