FINAL PROJECT REPORT
HOUSEHUNT: Finding your perfect rental home

Team ID: LTVIP2026TMIDS24425

INTRODUCTION

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. Here are some common features you might find in a house rent app

1.1 Project Overview

- The **House Rent App – House Hunt** is a full-stack web application built using the MERN stack (MongoDB, Express.js, React.js, and Node.js).
- This platform bridges the gap between renters and property owners, allowing for secure, real-time interaction. With an easy-to-use interface, it empowers users to browse, list, and manage properties, while administrators oversee activities for quality and compliance.

Key Highlights:

- Provides **role-based access** for Renters, Owners, and Admins, ensuring tailored dashboards and permissions for each user type.
- Enables **secure property listing, request handling, and booking** workflows, streamlining the traditional rental process.
- Offers **real-time status updates**, property management features, and renter-owner interaction tools to enhance platform usability.
- **Additional Point:**

  Supports **centralized rental management**, reducing the need for multiple platforms by combining search, booking, user verification, and communication into one integrated system.

  .

1.2 Purpose

The main purpose is to:

- Facilitate rental property discovery and management
- Provide real-time availability status
- Ensure security through user role verification (admin approval)
- Offer a scalable, modular architecture for future upgrades

2. IDEATION PHASE

2.1 Problem Statement

Renters often face difficulties such as outdated listings, lack of verification, and inefficient communication. On the other hand, property owners struggle to reach potential tenants efficiently. A unified platform is required to resolve these issues by providing transparency, automation, and effective data handling.

2.2 Empathy Map Canvas

HEAR

- "It's hard to trust online rental posts" — Renters are concerned about scams and fake listings.
- "Some owners don't respond" — Users often face poor communication after expressing interest.

SEE

- Disorganized listings with incomplete or inconsistent property details.
- Poor-quality or outdated property photos that fail to give a clear view of the house.

SAY & DO

- Visit multiple websites and apps to compare options, often leading to confusion.
- Contact many owners individually, leading to slow and uncoordinated communication.

PAIN

- The rental search process is time-consuming and repetitive across different platforms.
- High risk of scams or misleading listings due to lack of verification.

GAIN

- Access to verified property listings with accurate information and clear visuals.
- A secure platform with real-time updates and streamlined communication with owners.

2.3 Brainstormed Features

- Role-based access: Admin, Renter, Owner
- Approval mechanism for owner listing rights
- Property CRUD operations with booking system
- Email/push notification alerts for property status
- Virtual tours and high-resolution image galleries
- Integration with legal/mortgage experts for future scope

3. REQUIREMENT ANALYSIS

The requirement analysis for the *House Hunt* web application involves understanding user needs, system functionality, data flow, and the technologies required to build a robust, efficient, and user-friendly platform. It encompasses both functional and non-functional aspects that guide the project from design to deployment.

3.1 Customer Journey Map

The Customer Journey Map outlines each touchpoint that a user experiences while interacting with the House Hunt application. It helps us understand user behavior, emotions, goals, and potential friction points.

A customer journey map in the context of House Hunt captures the end-to-end experience of users — from their first interaction with the platform to the final booking confirmation. It begins when a user (either a renter or owner) visits the platform and registers through a secure sign-up process. Once logged in, renters are able to browse or search through the list of available properties using filters such as location, price, property type, and amenities.

Upon selecting a property, the renter sends a booking request by submitting a form with their details. The property owner receives this request and can approve or reject it. If approved, the renter sees an updated status indicating successful booking. In the background, the admin monitors all user activities, verifies new owner accounts, and ensures data integrity. This entire flow is designed to be seamless and intuitive, creating a smooth experience for every user.

Stage	**Action**	**Emotion**	**Opportunity**
1. Awareness	Renter/Owner visits site	Curious, hopeful	Show value with featured listings
2. Registration/Login	User signs up or logs in	Anxious, quick entry needed	Simple and fast login with JWT
3. Browsing Properties	Renter uses filters to search	Excited, selective	Offer advanced filters, save searches
4. Booking Request	Renter submits booking form	Uncertain, hopeful	Provide clear feedback and tracking
5. Owner Action	Owner approves/rejects	Responsible, cautious	Admin verification of both parties
6. Booking Confirmation	Status shown as booked	Satisfied, confident	Send notification and confirmation email
7. Admin Monitoring	Admin reviews all actions	Analytical	Dashboard insights and role approvals

3.2 Solution Requirements

The House Hunt platform requires both functional and non-functional capabilities to meet user expectations and ensure system performance.

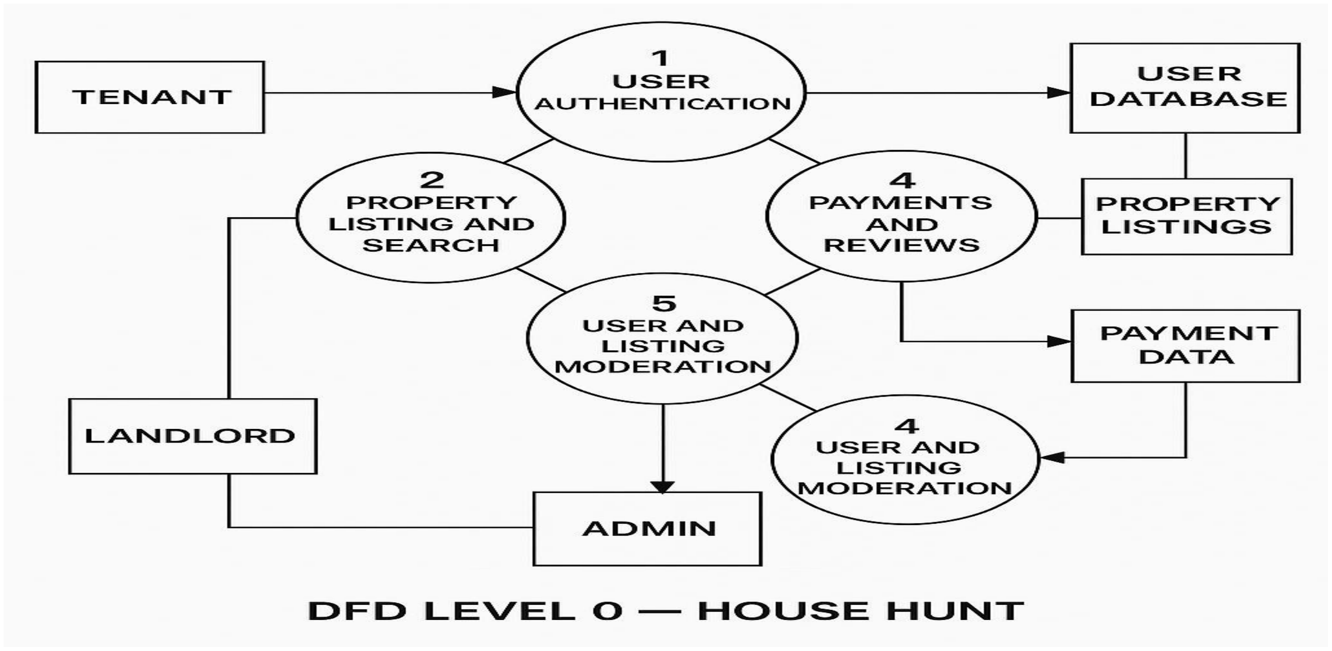
Functional Requirements

- Functionally, the system must support user registration and secure authentication for different roles — renters, property owners, and the admin.
- Once logged in, renters should be able to view all listed properties, search using dynamic filters, and initiate booking requests.
- Owners should be able to add, edit, and delete property listings, as well as manage incoming booking requests by approving or rejecting them.
- The admin plays a critical role in ensuring the integrity of the platform by approving new property owners and managing users and listings.
- Admins must have access to a dashboard where they can monitor user activities, verify ownership claims, and handle flagged properties or accounts.
- In addition to user-specific features, the system must handle property images, floor plans, virtual tours, and form submissions efficiently.
- Real-time status updates for bookings and properties are also part of the essential functional features.

Non-Functional Requirements

- In terms of non-functional requirements, usability is a top priority. The interface must be clean, responsive, and intuitive to ensure that users of all types — from tech-savvy owners to first-time renters — can navigate the platform with ease.
- Performance is equally critical; the application must deliver fast response times for actions such as property searches, booking submissions, and status updates.
- Security is fundamental, especially when handling user data and authentication. The platform should implement encrypted password storage, secure login tokens using JWT, and protected routes based on user roles.
- It should also provide data validation to prevent injection attacks or other vulnerabilities.
- Scalability is important for handling growth. As the number of users and property listings increases, the backend and database should scale efficiently without affecting speed or uptime. Reliability and availability must be maintained to ensure minimal downtime and maximum user trust.
- Maintainability is achieved by structuring the codebase into reusable modules and following industry-standard development practices. This makes it easier for future developers to understand, extend, or debug the system. Cross-platform compatibility ensures the platform performs consistently across major browsers and devices.

3.3 Data Flow Diagram (DFD)



A Data Flow Diagram (DFD) is a graphical representation of the flow of data within a system. It helps to visualize how information is input, processed, and output across different modules of the application. In the context of the *House Hunt* application, the DFD provides insight into how data moves between users (Renters, Owners, Admins) and the system's backend logic and database.

Level 0 DFD – Context Diagram

The Level 0 DFD, also known as the context-level diagram, provides a high-level overview of the system as a single process. It shows how external entities interact with the core system and what data flows between them.

In the House Hunt Level 0 DFD:

- Renters interact with the system to register, log in, view properties, and submit booking requests.
- Owners register, request admin approval, list properties, and manage booking responses.
- Admins access the system to verify users, approve owners, and monitor listings.
- The system communicates with a central MongoDB database to store and retrieve user, property, and booking data.

This level does not detail internal processes but gives a broad view of the entire platform's input/output boundaries.

Level 1 DFD – Process Breakdown

The Level 1 DFD decomposes the main system into individual sub-processes. It explains what operations are performed on the data received from users and how this data is stored or forwarded.

Key processes in the House Hunt Level 1 DFD include:

1. User Authentication Module
   - Handles login and registration.
   - Receives credentials and returns authentication tokens (JWT).
   - Stores hashed passwords and user details in the database.
2. Property Management Module
   - Allows owners to add, update, or delete properties.
   - Handles image uploads and property details (type, price, features).
   - Saves this data to the database for renters to view.
3. Property Browsing & Filtering Module
   - Provides renters with property listings based on filters like location, price, and availability.
   - Queries the database and returns relevant listings in real-time.
4. Booking Management Module
   - Accepts booking requests from renters and stores them with status as "Pending".
   - Allows owners to approve or reject bookings.
   - Updates booking status in the database.
5. Admin Control Module
   - Enables admin to view all users and property listings.
   - Approves or rejects owner registration requests.
   - Can deactivate properties or users violating terms.
6. Notification & Status Update Module
   - Sends booking confirmation or rejection notifications.
   - Displays current booking status to renters.
   - Provides real-time updates in the renter and owner dashboards.

Data Stores

The system stores all necessary data in a MongoDB database, structured into collections like:

- Users (storing details of renters, owners, admins)
- Properties (storing all property listings and details)
- Bookings (storing booking history and status)

Each module interacts with the data stores using Mongoose, ensuring data consistency and validation.

External Entities

- Renter: Accesses properties, submits booking requests.
- Owner: Adds/manages listings, updates booking status.
- Admin: Approves users, manages system activities.

These entities exchange data through HTTP requests handled by the Express.js backend.

3.4 Technology Stack

The House Hunt app uses the **MERN stack**, supported by various libraries and tools to enhance performance and user experience.

Frontend Technologies:

- React.js is used to build interactive UI components.
- Libraries like Bootstrap, Material UI, and Ant Design help with styling and layout.
- Axios is used to make API calls to the backend server.

Backend Technologies:

- Node.js serves as the runtime environment.
- Express.js manages routing and server logic.
- Multer is used to upload images (for property listings).
- JWT and bcryptjs are used for secure authentication and password encryption.

Database:

- MongoDB stores user profiles, property listings, and booking data.
- Mongoose is used as an ODM to simplify schema creation and data validation.

Other Tools and Libraries:

- Moment.js is used for date/time formatting in bookings.
- Nodemon helps in development by automatically restarting the server on changes.

- Git and GitHub are used for version control and team collaboration.
- Postman is used for API testing during development.

This technology stack provides a reliable, scalable, and secure foundation for the application.

4. PROJECT DESIGN

4.1 Problem-Solution Fit

The House Hunt platform was conceptualized in response to the fragmented and often unreliable process of renting properties online. Traditional methods lack real-time updates, are prone to fraudulent listings, and don't provide transparent communication between renters and property owners.

This project ensures a **strong problem–solution fit** by directly addressing those gaps through:

- **Real-time property availability** and booking status
- **Admin verification mechanisms** to build trust and accountability
- **Clear role separation** between renters, owners, and administrators
- **Modern UI/UX** design that improves accessibility and navigation
- **Centralized listing management** for owners to track properties, bookings, and availability

The solution ensures that both user needs and technical feasibility are met while creating a seamless digital rental experience.

4.2 Proposed Solution

The proposed solution is a full-stack web application that provides a central portal where renters can find and book properties, owners can manage their listings, and admins can oversee all platform activities. The system offers tailored dashboards for each role and ensures secure, real-time interactions between them.

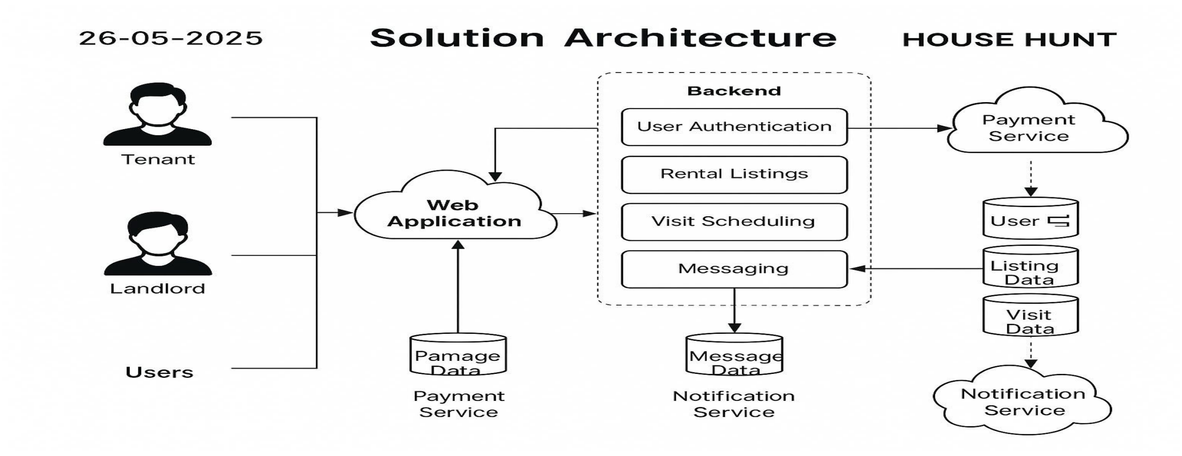
Key highlights of the proposed system include:

- **Role-based Dashboards**: Renters see available properties and track bookings; owners manage property listings and booking responses; admins manage users and listings.
- **Dynamic Property Listings**: Properties can be added with images, amenities, location, and availability status. Filters are applied for smarter search.
- **Booking Workflow**: Renters initiate a booking; owners approve/reject it; the system reflects the status in real time.
- **Admin Approval System**: To maintain quality, only admin-approved owners can list properties. Admins also handle platform moderation.
- **Secure Authentication**: JWT-based login system for safe, tokenized sessions and route protection.

- **Responsive UI**: Built using React.js and styled with Ant Design and Bootstrap for a smooth user experience across devices.

The proposed system aims to simplify the rental experience while offering transparency, trust, and control to all users involved.

4.3 Solution Architecture



The architecture of the House Hunt application follows a **modular, client-server model** with a separation of concerns across the frontend, backend, and database layers.

Frontend (Client-Side)

- Developed in **React.js**, the frontend handles user interactions and UI rendering.
- It communicates with the backend using **Axios** to send/receive data via REST APIs.
- Routing and protected views are managed within React using route guards for authorized access.
- Libraries like **Material UI**, **Ant Design**, and **Bootstrap** are used to maintain a clean, responsive layout.

Backend (Server-Side)

- The backend is powered by **Node.js** and **Express.js**.
- It manages all business logic, route handling, user authentication, and API endpoints.
- Secure login and signup are handled with **JWT (jsonwebtoken)**, and passwords are encrypted with **bcryptjs**.
- It also supports file uploads (e.g., property images) using **Multer**.

Database (MongoDB)

- All application data (users, properties, bookings) is stored in **MongoDB**.
- **Mongoose** is used for schema modeling, validation, and database operations.
- Each document corresponds to a data entity (e.g., a property or a booking).

Data Flow and Communication

- The React frontend sends HTTP requests to the Express backend.
- The backend processes the request, performs database queries using Mongoose, and sends back the appropriate response.
- JWT tokens are used to maintain session security and verify the identity of logged-in users.

This architecture is chosen for its **scalability**, **security**, and **real-time performance** capabilities, ensuring that the platform remains responsive and functional even as the user base grows.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The *House Hunt* project follows the **Agile Scrum methodology**, a flexible and iterative approach to software development. It emphasizes collaborative effort, rapid delivery, and continuous feedback to adapt to changing requirements efficiently.

Product Backlog and Sprint Planning:

ID	**User Story / Feature**	**Role**	**Goal / Action**	**Benefit / Purpose**	**Priority**
US01	Register/Login/Logout	User	Sign up and log in/out securely	Access personal dashboard securely	High
US02	Forgot Password	User	Recover password	Regain access to account	High
US03	View Property Listings	Renter	Browse/search/filter property listings	Find properties matching preferences	High
US04	Add Property	Owner	Add new property details	Offer for rent/sale	High
US05	Edit/Delete Property	Owner	Modify or remove listed properties	Keep listings up to date	Medium
US06	View Property Details	Renter	View detailed info of a selected property	Help make decisions	High
US07	Book a Property	Renter	Submit booking request with personal info	Schedule viewing or rent	High
US08	View Booking History	Renter	Track my previous or ongoing bookings	Manage personal booking records	Medium

US09	View Incoming Bookings	Owner	View renters interested in my properties	Manage bookings	Medium
US10	Grant Owner Access	Admin	Approve/not approve owner registration	Control property listing access	High
US11	View All Users	Admin	Access all user info	Monitor and manage users	High
US12	View All Bookings	Admin	Monitor all platform bookings	Ensure transparency and activity tracking	Medium
US13	Search Filter	Renter	Apply location/type filters while browsing	Quickly find relevant listings	High
US14	Upload Property Images	Owner	Upload property images	Help renters visualize property	Medium
US15	Role-Based Routing	System	Navigate user to their respective dashboard	Provide personalized navigation	High

The development cycle is divided into four sprints, each targeting specific user stories from the product backlog:

- **Sprint t-1** focused on implementing core **user authentication**.
- **Sprint t-2** introduced **property search and booking** functionalities.
- **Sprint t-3** prioritized **landlord features, user profiles, and messaging**.
- **Sprint t-4** completed the system with **payment integration, reviews, and profile security**.

Each user story was assigned **story points** based on complexity and **priority** based on business value. This structure ensured clarity in development goals and efficient time allocation.

Sprint Execution and Tracking

The project employed a detailed sprint tracker to monitor progress:

- Each sprint lasted **6 days** with a consistent story point target of **20 points**.
- Tracking included metrics such as planned vs. actual release dates and completed story points.
- The **burndown chart** visualized sprint progress, helping stakeholders quickly assess velocity and productivity.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

House Hunt is an intuitive and reliable house rental platform that helps tenants easily find and rent homes while enabling landlords to list and manage properties. Key features include:

- Verified property listings
- Advanced search filters (location, budget, amenities, etc.)
- Property visit booking
- Secure digital agreements
- Reviews and ratings
- Admin panel for user and listing moderation

Testing Scope:

- Tenant sign-up and login
- Landlord registration and listing creation
- Search and filter functionality
- Visit scheduling module
- Digital agreement upload/signing
- Admin controls (user management, content moderation)

Requirements to be tested

As a tenant, I want to search and schedule property visits easily.

As a landlord, I want to list and manage rental properties.

- As a user, I want a secure platform with chat, digital agreements, and payments.
- As an admin, I want control over listings, users, and feedback.

Testing environment:

Credentials:

- **Tenant:** test.tenant@example.com / tenant123
- **Landlord:** test.landlord@example.com / landlord123
- **Admin:** admin@househunt.com / admin pass

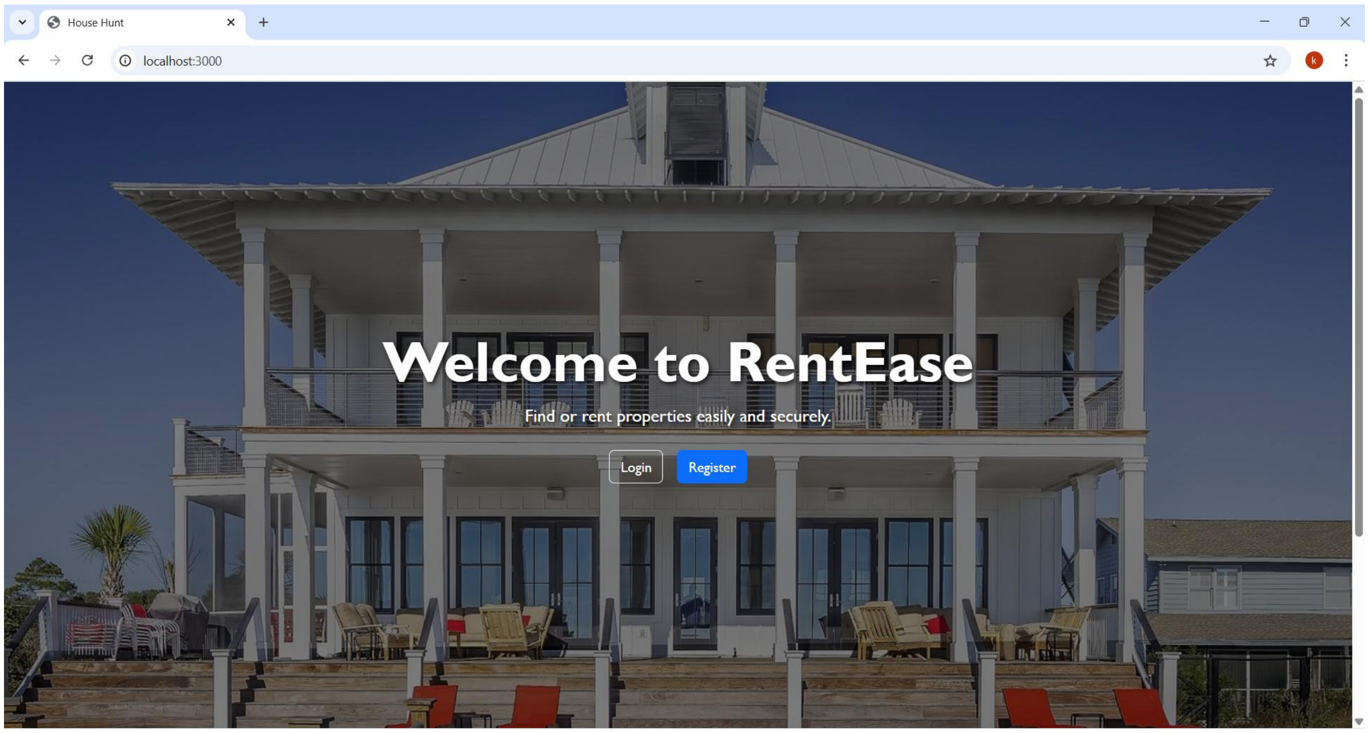
Testcases:

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
TC-001	User Registration	1. Visit site 2. Click "Sign Up" 3. Fill & submit form	Profile saved	Account created, redirected to dashboard	[Pass/Fail]
TC-002	Landlord Listing Creation	1. Login as landlord 2. Add property details 3. Submit	Listing visible in search results	Account created, redirected to dashboard	[Pass/Fail]
TC-003	Property Search & Filters	1. Login as tenant 2. Use search & filters	Only matching listings displayed	Appointment will be booked	[Pass/Fail]

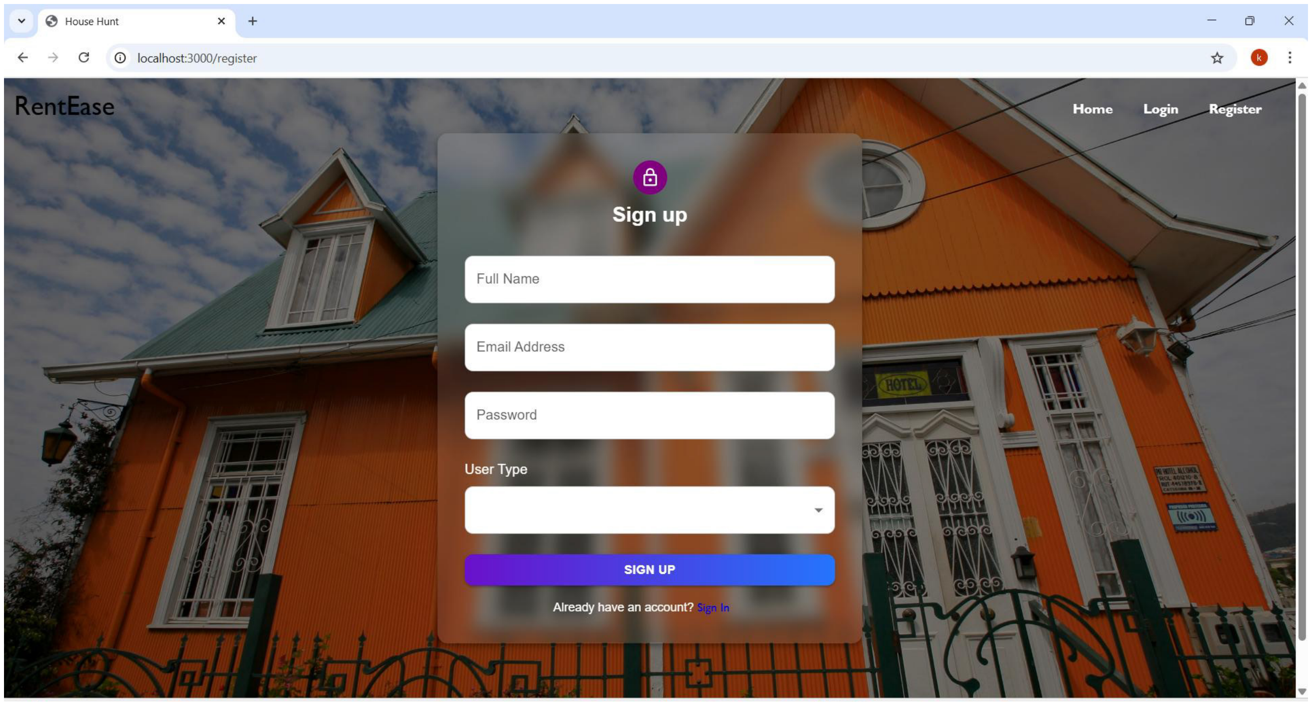
7. RESULTS

7.1 Output Screenshots

❖**Homepage:**

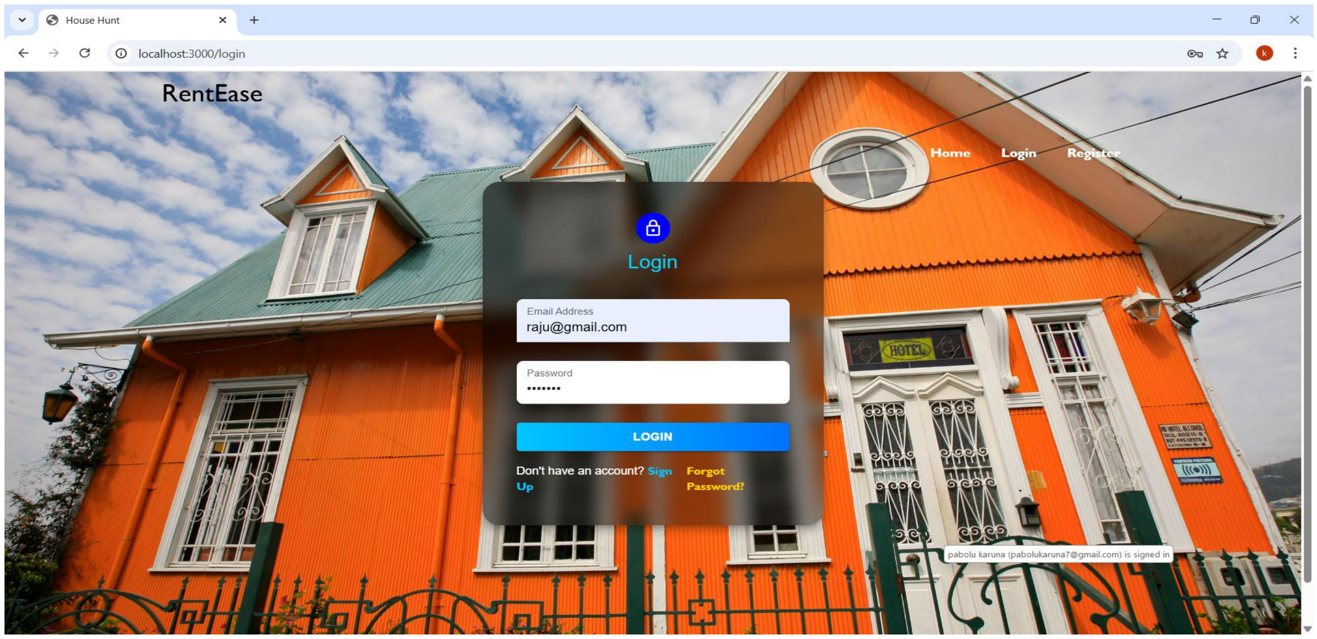


❖Registration page:



❖

❖Login page:



House Hunt

localhost:3000/login

RentEase

Home Login Register

Login

Email Address: raju@gmail.com

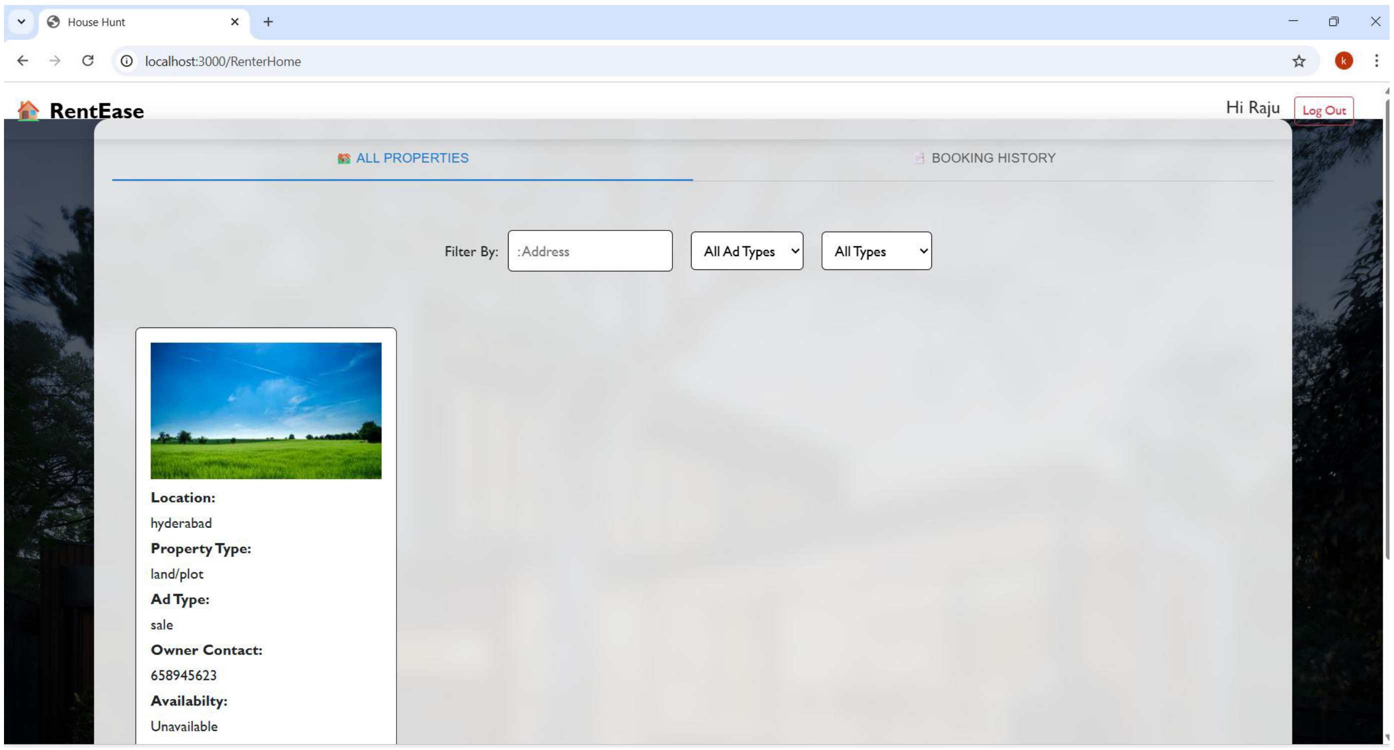
Password: •••••••

LOGIN

Don't have an account? Sign Up Forgot Password?

pabolu karuna (pabolukaruna7@gmail.com) is signed in

❖Renter Home page:



House Hunt

localhost:3000/RenterHome

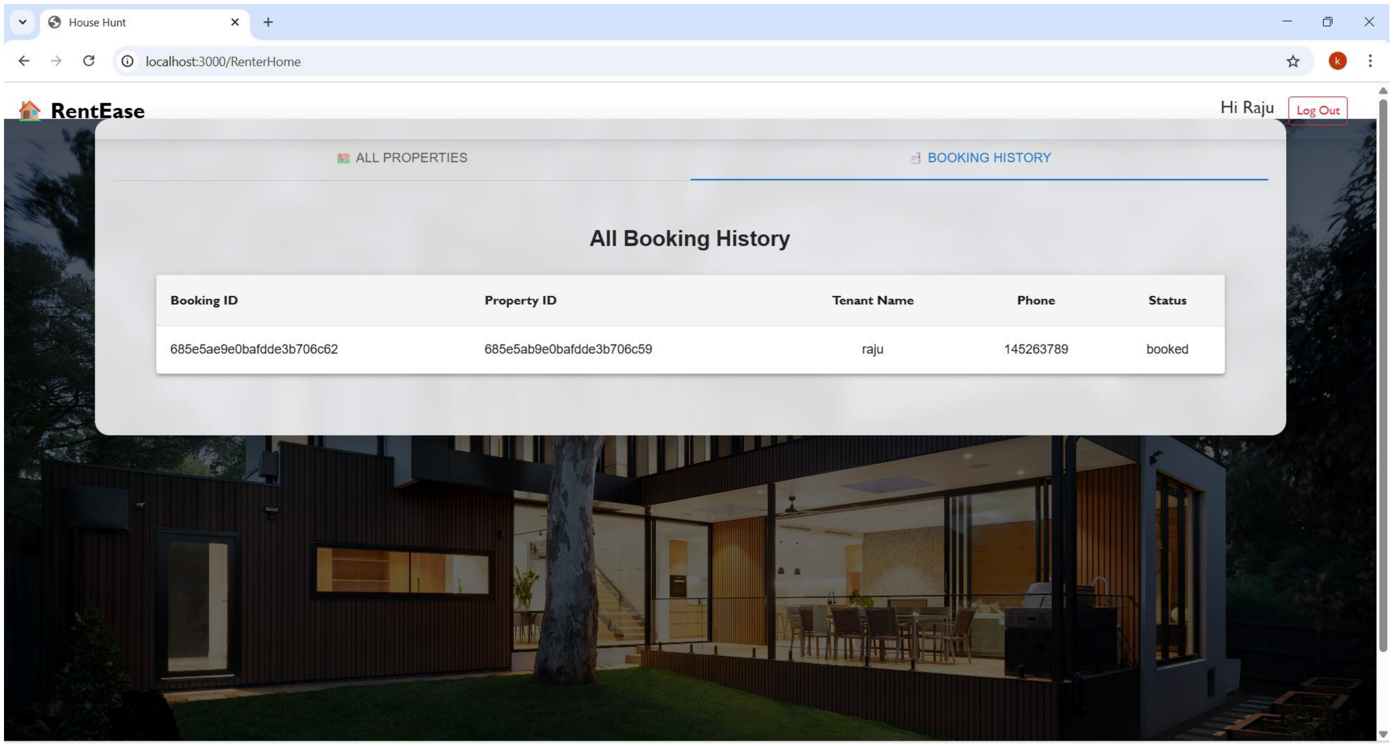
RentEase Hi Raju Log Out

ALL PROPERTIES BOOKING HISTORY

Filter By: :Address | All Ad Types | All Types

Location:
hyderabad
Property Type:
land/plot
Ad Type:
sale
Owner Contact:
658945623
Availabilty:
Unavailable

❖Renter Booking history page:



House Hunt

localhost:3000/RenterHome

RentEase

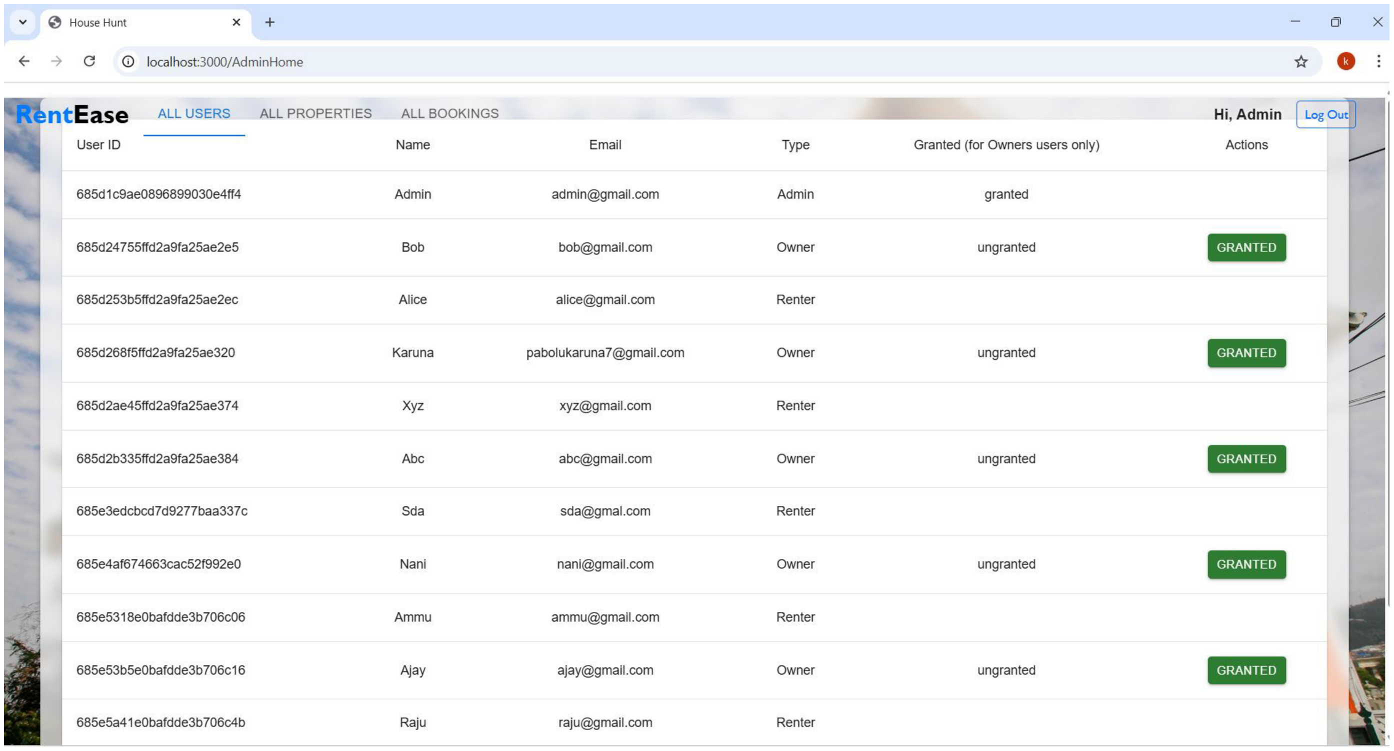
Hi Raju Log Out

ALL PROPERTIES BOOKING HISTORY

All Booking History

Booking ID	Property ID	Tenant Name	Phone	Status
685e5ae9e0bafdde3b706c62	685e5ab9e0bafdde3b706c59	raju	145263789	booked

❖Admin home page:



House Hunt

localhost:3000/AdminHome

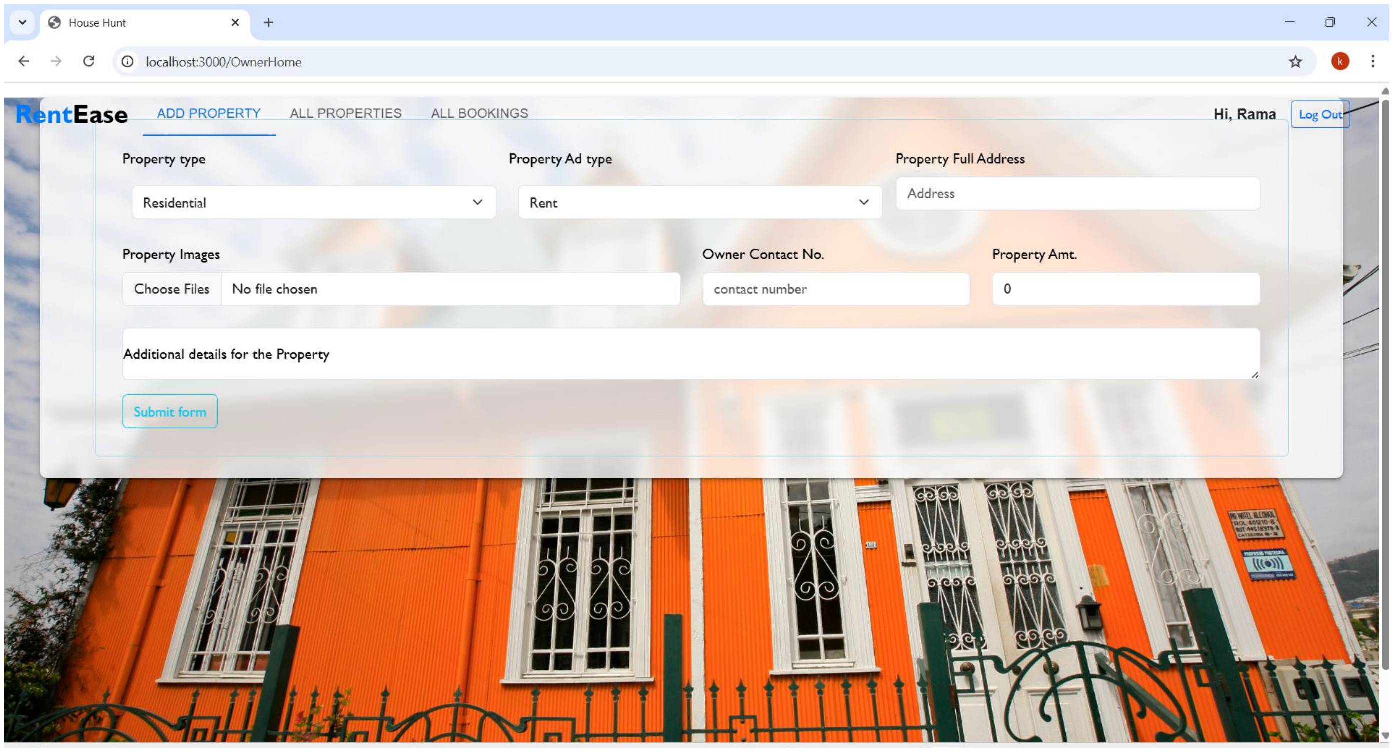
RentEase

ALL USERS ALL PROPERTIES ALL BOOKINGS

Hi, Admin Log Out

User ID	Name	Email	Type	Granted (for Owners users only)	Actions
685d1c9ae0896899030e4ff4	Admin	admin@gmail.com	Admin	granted	
685d24755ffd2a9fa25ae2e5	Bob	bob@gmail.com	Owner	ungranted	GRANTED
685d253b5ffd2a9fa25ae2ec	Alice	alice@gmail.com	Renter		
685d268f5ffd2a9fa25ae320	Karuna	pabolukaruna7@gmail.com	Owner	ungranted	GRANTED
685d2ae45ffd2a9fa25ae374	Xyz	xyz@gmail.com	Renter		
685d2b335ffd2a9fa25ae384	Abc	abc@gmail.com	Owner	ungranted	GRANTED
685e3edcbcd7d9277baa337c	Sda	sda@gmal.com	Renter		
685e4af674663cac52f992e0	Nani	nani@gmail.com	Owner	ungranted	GRANTED
685e5318e0bafdde3b706c06	Ammu	ammu@gmail.com	Renter		
685e53b5e0bafdde3b706c16	Ajay	ajay@gmail.com	Owner	ungranted	GRANTED
685e5a41e0bafdde3b706c4b	Raju	raju@gmail.com	Renter		

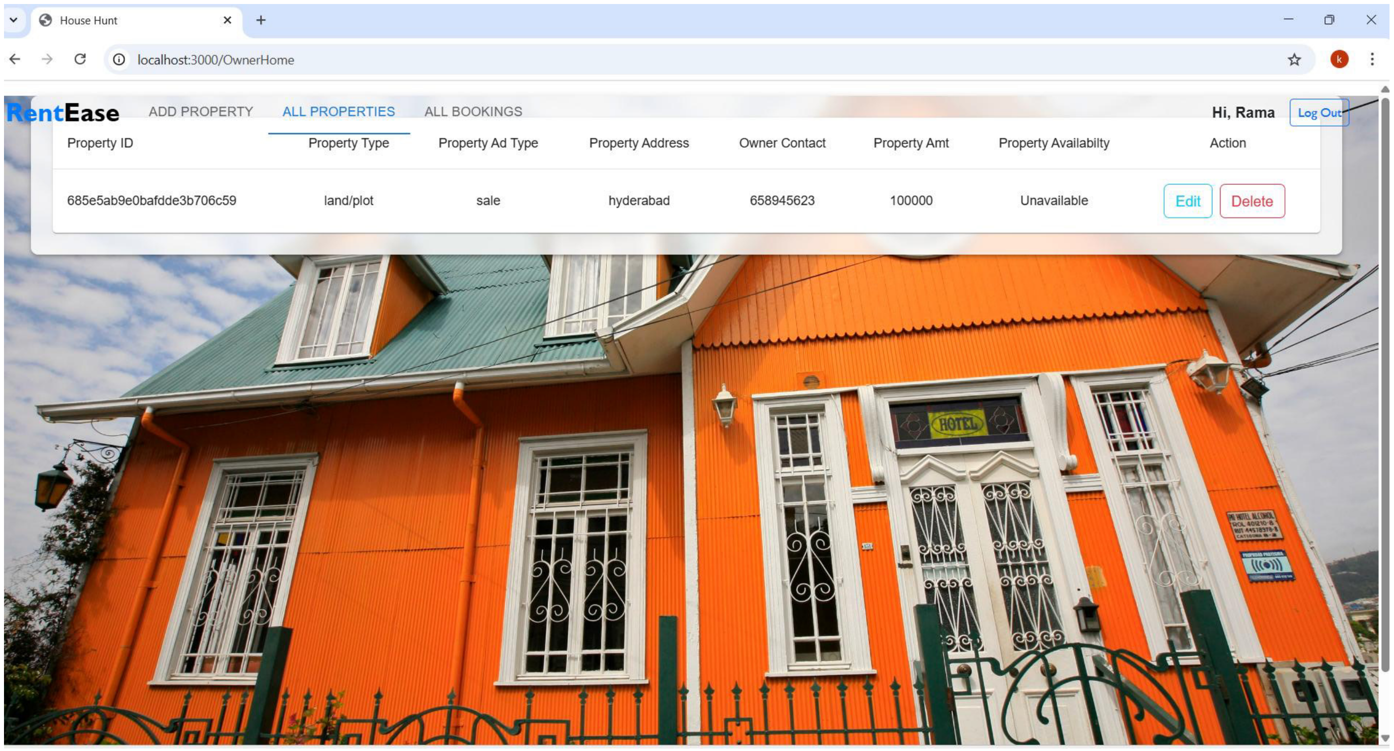
❖Owner home page:



Property type	Property Ad type	Property Full Address
Residential	Rent	Address
Property Images	Owner Contact No.	Property Amt.
Choose Files No file chosen	contact number	0

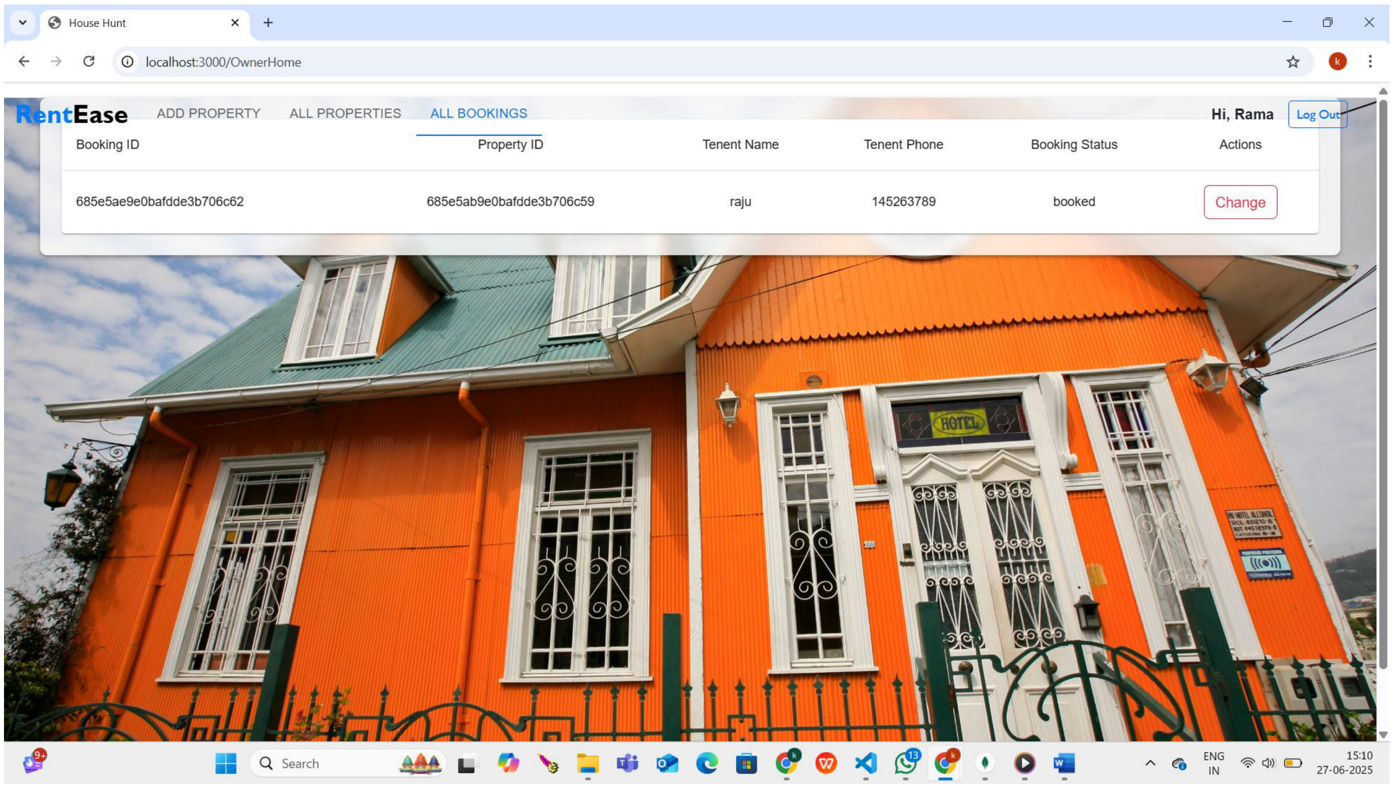
Additional details for the Property

❖**All properties page:**



Property ID	Property Type	Property Ad Type	Property Address	Owner Contact	Property Amt	Property Availabilty	Action
685e5ab9e0bafdde3b706c59	land/plot	sale	hyderabad	658945623	100000	Unavailable	Edit Delete

❖**All bookings page:**



Booking ID	Property ID	Tenent Name	Tenent Phone	Booking Status	Actions
685e5ae9e0bafdde3b706c62	685e5ab9e0bafdde3b706c59	raju	145263789	booked	Change

8.ADVANTAGES & DISADVANTAGES

- **Simplified Rental Process**
  Provides a one-stop solution for property listing, renting, and tenant management.
- **Role-Based Access**
  Features separate dashboards for Admins, Owners, and Renters with tailored functionality.
- **Scalable Architecture**
  Built on the MERN (MongoDB, Express.js, React.js, Node.js) stack for performance and scalability.
- **Real-Time Property Updates**
  Owners can list or remove properties instantly; renters get live availability.
- **Secure Authentication**
  JWT and bcrypt ensure secure login and registration flows for all users.

Disadvantages

- **Requires Internet Connection**
  All features depend on online access, limiting use in offline scenarios.

- **Initial Development Complexity**
  Developing and managing role-based routing and UI was challenging.
- **Limited Mobile Optimization**
  Currently optimized for desktops; mobile experience may need improvement.
- **Maintenance Overhead**
  Admin needs to constantly monitor listings, users, and rental activities.

9.CONCLUSION

House Hunt is a modern web-based rental application developed using the MERN stack, designed to streamline the process of house renting for both property owners and tenants. It acts as a bridge between renters seeking accommodation and owners offering rental spaces. The system is robust, user-friendly, and role-based, offering three separate user experiences:

- **Renters** can browse, request, and manage their bookings.
- **Owners** can list properties, manage requests, and monitor engagement.
- **Admins** can oversee platform health, user control, and content moderation. From secure user authentication to dynamic property listings, the platform embodies the complete workflow of a digital real estate rental solution.

Key Achievements

- **Multi-Role Authentication System**
  - Implemented JWT-based login/registration with hashed passwords for Renters, Owners, and Admins.
- **Property Listing & Booking Management**
  - Owners can add/update/delete properties with image support; renters can request/book listed properties.
- **Admin Dashboard**
  - Admins can view all registered users, manage platform data, and monitor activities across the application.
- **Interactive UI with Dynamic Content Rendering**
  - Role-based dashboards render personalized data using protected routes and React state/context APIs.
- **Database Integration & RESTful APIs**
  - Built a backend that handles CRUD operations for users, properties, and requests using Express.js and MongoDB.

10. FUTURE SCOPE

1. **Chat and Messaging**
   - Real-time conversations between renters and owners.
2. **Verified Profiles and Listings**
   - Verification badges for trusted users and property owners.

3. **Search Filters and Recommendations**
    - Location-based and price-based filters with ML-driven recommendations.
4. **Third-Party Integration**
    - Integration with government ID verification APIs, payment systems, and calendar booking tools.
5. **Analytics Dashboard for Owners/Admins**
    - Insights into most viewed properties, booking trends, and user engagement.
6. **Multilingual Support**
    - Make the platform more accessible to users from various regions by supporting local languages.

11. APPENDIX

- **Source Code:** **https://drive.google.com/drive/folders/10OstrbGEPKtXDENGkerKBShejPJIbVgj?usp=sharing**
- **Project Demo Link:**

  https://drive.google.com/file/d/1PqJBucNH4oAnXBrWI3Rk4_Dw7iDupMlg/view?usp=sharing