Full Stack Development with MERN

HouseHunt: Finding Your Perfect Rental Home

1. Introduction

This document provides a comprehensive overview of **HouseHunt: Finding Your Perfect Rental Home**, a full-stack rental housing application built using the MERN (MongoDB, Express.js, React.js, Node.js) stack. It details the project's purpose, architecture, setup instructions, and key features, serving as essential documentation for developers, stakeholders, and future contributors.

1.1. Project Title

HouseHunt: Finding Your Perfect Rental Home

1.2. Team Members

- Team Leader: **Kalidindi Naga Surya Uma Siva Rama Raju**
- Team Member: **Harshitha Valli Modugumudi**

1.3. Team ID

LTVIP2025TMID54974

---

2. Project Overview

2.1. Purpose

HouseHunt is designed to simplify the rental property search and management process. Its primary purpose is to create a user-friendly platform that seamlessly connects renters with their ideal homes and empowers property owners to efficiently manage their listings. The application aims to streamline the entire rental journey, from property discovery to agreement handling, fostering a hassle-free experience for all parties involved.

2.2. Features

The HouseHunt application comes equipped with a robust set of features to facilitate a complete rental ecosystem:

- **Property listings with images and descriptions:** Detailed profiles for each property, including high-quality visuals and comprehensive information.

- **Advanced search filters (location, price, bedrooms, etc.):** Empowers renters to refine their search based on specific criteria, ensuring they find properties that perfectly match their needs.
- **User registration as Renter or Owner:** Supports distinct user roles with tailored functionalities for each type of user.
- **Inquiry and booking flow:** A clear, guided process for renters to inquire about properties and initiate booking requests.
- **Owner property management dashboard:** A dedicated interface for property owners to add, update, and oversee their listings, as well as manage inquiries.
- **Admin verification and governance:** An administrative panel to oversee the platform, approve new properties, manage users, and ensure content quality.
- **In-app messaging for negotiations:** Facilitates direct communication between renters and owners for discussions and negotiations.
- **Transaction and digital lease agreement handling:** A framework to manage financial transactions and generate digital lease agreements (future enhancement).

---

3. Architecture

HouseHunt leverages the power of the MERN stack to provide a scalable and efficient full-stack solution.

3.1. Frontend

The user-facing part of HouseHunt is built with **React.js**, a popular JavaScript library for building dynamic user interfaces.

- **Styling:** The frontend utilizes **Bootstrap** for responsive layouts and **Material UI** for a modern, component-based design, ensuring a consistent and appealing user experience.
- **API Interaction: Axios** is used for making asynchronous HTTP requests to communicate with the backend APIs, fetching and sending data securely and efficiently.

3.2. Backend

The server-side logic and API services are developed using **Node.js** and the **Express.js** framework.

- **RESTful APIs:** The backend provides a set of **RESTful APIs** for all CRUD (Create, Read, Update, Delete) operations, allowing the frontend and other services to interact with the database.
- **Authentication and Booking Management:** It includes dedicated routes for user authentication (login, registration) and robust logic for managing property bookings, ensuring secure and orderly transactions.

3.3. Database

MongoDB, a NoSQL database, serves as the primary data store for HouseHunt.

- **Data Storage:** MongoDB is ideal for storing diverse data types such as user details, property listings with rich attributes, booking information, and message logs. Its flexible schema allows for efficient data management and scalability.
- **Collections:** The main collections include Users (for renter and owner profiles), Properties (for all listed homes), Bookings (for rental transactions), and Messages (for in-app communications).

---

4. Setup Instructions

To get the HouseHunt application up and running on your local machine, follow these steps.

4.1. Prerequisites

Before you begin, ensure you have the following software installed on your system:

- Node.js >= 16: The JavaScript runtime environment required for both frontend and backend.
- MongoDB: Either installed locally on your machine or accessible via a cloud service like MongoDB Atlas.
- Git: For cloning the project repository.

4.2. Installation

1. Clone the repository: Open your terminal or command prompt and execute the following command to download the project files:

   ```
   git clone https://github.com/Siva-kalidindi/househunt.git
   ```

2. Navigate into the project: Change your current directory to the newly cloned project folder:
   cd househunt
3. Install dependencies: Install all required Node.js packages for both the client and server:

   ```
   npm install
   ```

4. Setup .env with necessary environment variables: Create a .env file in the server directory and configure essential variables such as your MongoDB connection URI

(DB_URI) and a JWT secret key (JWT_SECRET).
Example .env file content:

```
DB_URI=mongodb://localhost:27017/househunt_db
JWT_SECRET=your_super_secret_jwt_key
```

5. Start frontend and backend: Open two separate terminal windows. In the first window, navigate to the client directory and start the React development server. In the second window, navigate to the server directory and start the Node.js backend server.

```
# Terminal 1 (for frontend)
cd client
npm start
```

```
# Terminal 2 (for backend)

 cd server

 npm start
```

---

5. Folder Structure

Understanding the project's folder structure is crucial for navigation and development.

5.1. Client (React)

The client directory houses the React.js frontend application.

- /components: Contains reusable UI components such as Navbar, property Listings, and Filters.
- /pages: Holds the main views or screens of the application, like Home, Login, Register, and user/owner Dashboard.
- /services: Manages **Axios** API calls, encapsulating the logic for communicating with the backend.

5.2. Server (Node.js + Express)

The server directory contains the Node.js and Express.js backend application.

- /routes: Defines the API route files, separating concerns for authentication (auth), bookings (bookings), and properties (properties).

- /models: Contains **Mongoose schemas** for database collections, defining the structure for User, Property, and Booking data.
- /controllers: Implements the business logic for each API endpoint, handling requests and responses.
- /middlewares: Stores middleware functions, including authentication middleware for token verification and error handlers.

---

6. Running the Application

Once the setup is complete and dependencies are installed, you can launch the application.

6.1. Frontend

To start the React development server for the frontend, navigate to the frontend directory and run:

```bash
cd frontend
npm start
```

This will typically open the application in your default web browser at http://localhost:3000.

6.2. Backend

To start the Node.js Express server for the backend, navigate to the backend directory and run:

```bash
cd backend
npm start
```

The backend server will usually run on http://localhost:5000 (or another port specified in your .env file).

---

7. API Documentation

The backend exposes a set of RESTful APIs for interacting with the application's data.

7.1. Endpoints

Key API endpoints include:

- `POST /auth/signup`: Registers a new user account (Renter, Owner, or Admin).
- `POST /auth/login`: Authenticates a user and returns a JSON Web Token (JWT) for subsequent authenticated requests.
- `GET /general/properties`: Retrieves a list of available rental properties, with optional filters.
- `POST /general/properties`: Allows authenticated property owners to add a new property listing.
- `POST /general/bookings`: Facilitates the booking of a property by an authenticated renter.

7.2. Token Verification and Role-Based Access Control

Each protected endpoint rigorously includes **token verification** to ensure that only authenticated users can access them. Furthermore, **role-based access control** is implemented, meaning certain operations (like adding a property or accessing the admin panel) are restricted to users with specific roles (e.g., Owner, Admin).

---

8. Authentication

Authentication in HouseHunt is designed to be secure, stateless, and flexible.

8.1. JWT-Based Authentication

The application utilizes **JSON Web Token (JWT)-based authentication** for secure user login and protection of sensitive routes. Upon successful login, a JWT is issued to the user.

8.2. Token Storage

For convenience and persistence, the JWT is securely **stored in the client's local storage**. This allows users to remain logged in across browser sessions.

8.3. Role-Based Access

HouseHunt supports distinct user roles: **Admin**, **Owner**, and **Renter**. Each role has specific permissions and access levels within the application, ensuring that users can only perform actions relevant to their role.

8.4. Auth Middleware

A dedicated **authentication middleware** is implemented on the backend. This middleware intercepts incoming requests to protected routes, validates the JWT, and verifies the user's role before allowing access. This robust system ensures data integrity and security.

---

9. User Interface

The user interface of HouseHunt is designed for intuition and ease of use, catering to both renters and owners.

9.1. Home Page

The **Home page** provides a welcoming entry point with a prominent search bar and various filter options, allowing users to quickly begin their property search.

9.2. Listings Display

Property listings are elegantly displayed in a **card format**, each showcasing key information like property images, title, location, price, and a brief description, making it easy to browse.

9.3. Renter Dashboard

The **Renter dashboard** serves as a centralized hub for renters to view and track their inquiries, manage their current and past bookings, and communicate with property owners.

9.4. Owner Dashboard

The **Owner dashboard** offers a comprehensive set of tools for property owners to manage their listings, including options to add new properties, update existing ones, and respond to renter inquiries.

9.5. Admin Panel

The **Admin panel** provides administrators with a powerful interface to oversee the platform, including managing users, verifying property listings, and handling pending approvals, ensuring the smooth operation of HouseHunt.

---

10. Testing

Testing is a critical and continuous phase in software development, ensuring the reliability, functionality, and performance of the application. For HouseHunt, a robust testing strategy is essential to guarantee a seamless experience for both renters and property owners, and to identify and rectify defects early in the development lifecycle. This section outlines the planned approach to testing the various components of the MERN stack application.

10.1. Testing Methodologies and Tools

The testing strategy for HouseHunt will encompass various methodologies and utilize industry-standard tools to achieve comprehensive coverage and ensure the quality of the application. While specific test cases and detailed reports will be added as testing efforts progress, the core approaches include:

- **Unit Testing:**

    - **Purpose:** To test individual components or functions in isolation to ensure they perform as expected. This focuses on the smallest testable parts of the application.
    - **Frontend (React.js):** Jest, often combined with React Testing Library, will be used to thoroughly test individual React components. This includes verifying that UI elements render correctly, user interactions trigger the right functions, and data is displayed accurately based on props or state.
    - **Backend (Node.js/Express.js):** Jest or Mocha with Chai will be employed to test individual API routes, controller functions, and utility modules. This ensures that business logic, data processing, and error handling within each isolated unit of code function correctly.
- **Integration Testing:**
    - **Purpose:** To verify that different modules or services of the application work together correctly when integrated. This checks the interfaces and interactions between distinct components.
    - **Frontend-Backend Integration:** Tools like Cypress or Playwright will be considered for end-to-end (E2E) testing. These tools simulate realistic user flows through the UI to ensure the frontend correctly communicates with the backend APIs and displays appropriate responses, mimicking a user's complete journey through the application.
    - **API Integration:** Postman or Newman (Postman's command-line runner) will be used to test the interaction between various backend API endpoints. This ensures correct data flow and authentication across different services within the backend, verifying that APIs can successfully exchange information.
- **API Testing:**
    - **Purpose:** To test the functionality, reliability, performance, and security of the API endpoints independently of the user interface.
    - **Tools:** Postman will be a primary tool for manually testing RESTful APIs during development. For automation, these tests can be converted to automated scripts and run using Newman or integrated into broader test suites. This crucial step ensures that all endpoints for user registration, property listings, bookings, and authentication respond correctly with valid data and appropriate HTTP status codes under various conditions.
- **User Acceptance Testing (UAT):**
    - **Purpose:** To ensure that the application meets the defined business requirements and is suitable for its intended end-users in a real-world scenario.
    - **Process:** Key stakeholders, including potential renters and property owners, will be involved in UAT. Their feedback will be invaluable for validating usability, identifying any gaps in functionality from a user's perspective, and ensuring overall satisfaction with the application.
- **Performance Testing (Future Consideration):**
    - **Purpose:** To assess the application's responsiveness, stability, scalability, and resource usage under various load conditions.

    - **Tools:** In later stages of development, tools like JMeter or LoadRunner might be considered to simulate concurrent users and measure response times, throughput, and server resource utilization, ensuring the application can handle expected user traffic.
- **Test Coverage and Automated Pipelines:**
    - **Test Coverage:** A key objective will be to achieve a high percentage of code coverage through unit and integration tests. Tools integrated with Jest or other testing frameworks will help monitor and report on these metrics, guiding further testing efforts.
    - **Automated Testing Pipelines:** As the project matures, automated testing will be integrated into the Continuous Integration/Continuous Deployment (CI/CD) pipeline. This setup ensures that tests automatically run whenever code changes are committed, providing immediate feedback on the health of the codebase and helping to maintain a high standard of quality before any new features are deployed to production.

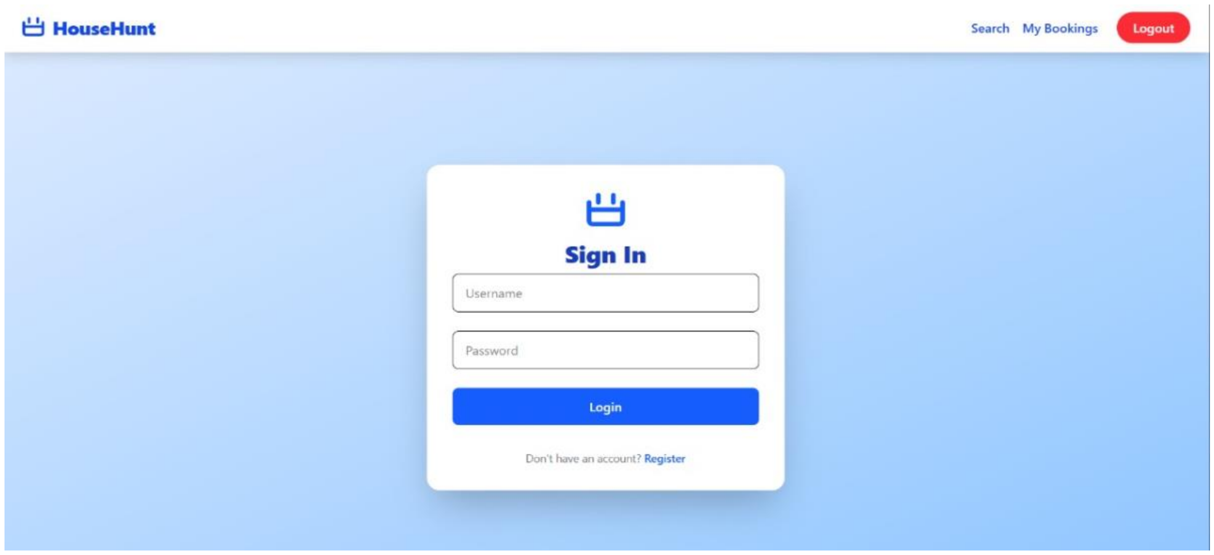
---

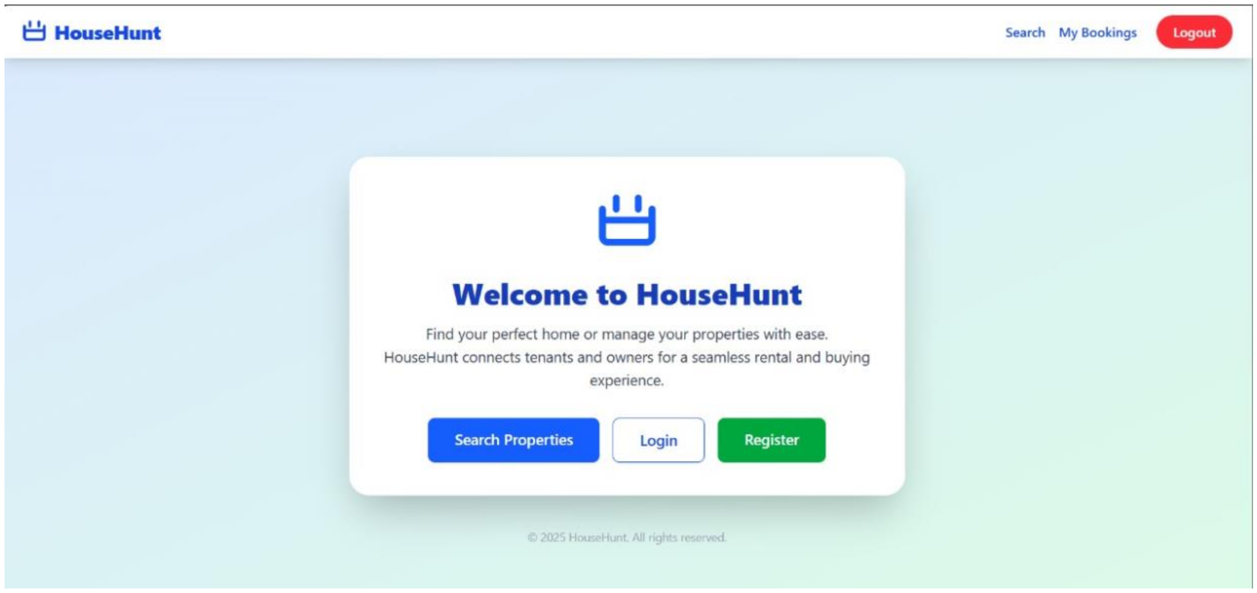
11. Screenshots or Demo

Visual aids are indispensable for understanding the application's user experience and functionality.

11.1. Images

- **Home Page with property listings:** A view showcasing the main landing page with search options and example property cards.



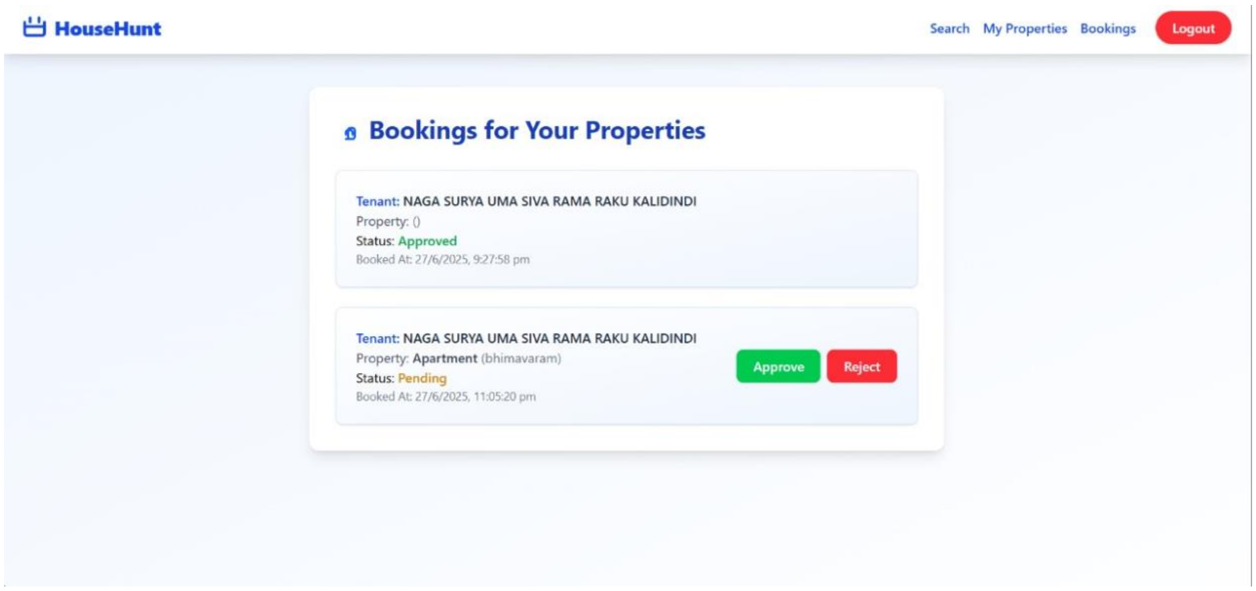


Welcome to HouseHunt

Find your perfect home or manage your properties with ease. HouseHunt connects tenants and owners for a seamless rental and buying experience.

© 2025 HouseHunt. All rights reserved.

- **Renter dashboard with booking status:** An illustration of a renter's personalized dashboard, highlighting current and past bookings with their respective statuses.

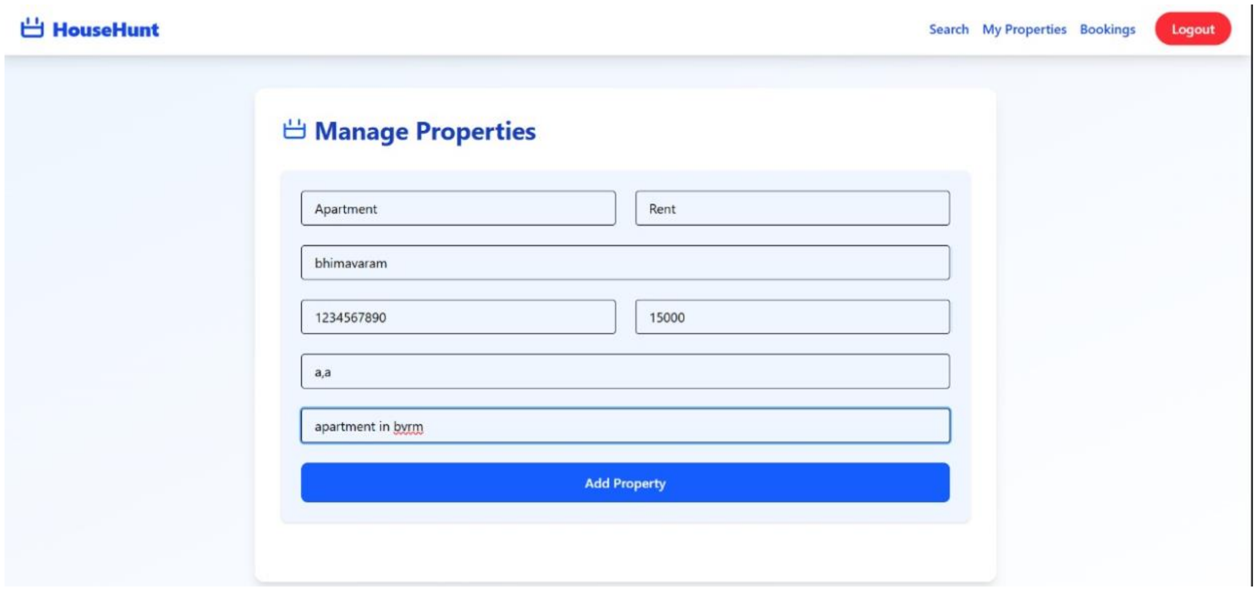


Bookings for Your Properties

Tenant: NAGA SURYA UMA SIVA RAMA RAKU KALIDINDI
Property: ()
Status: Approved
Booked At: 27/6/2025, 9:27:58 pm

Tenant: NAGA SURYA UMA SIVA RAMA RAKU KALIDINDI
Property: Apartment (bhimavaram)
Status: Pending
Booked At: 27/6/2025, 11:05:20 pm

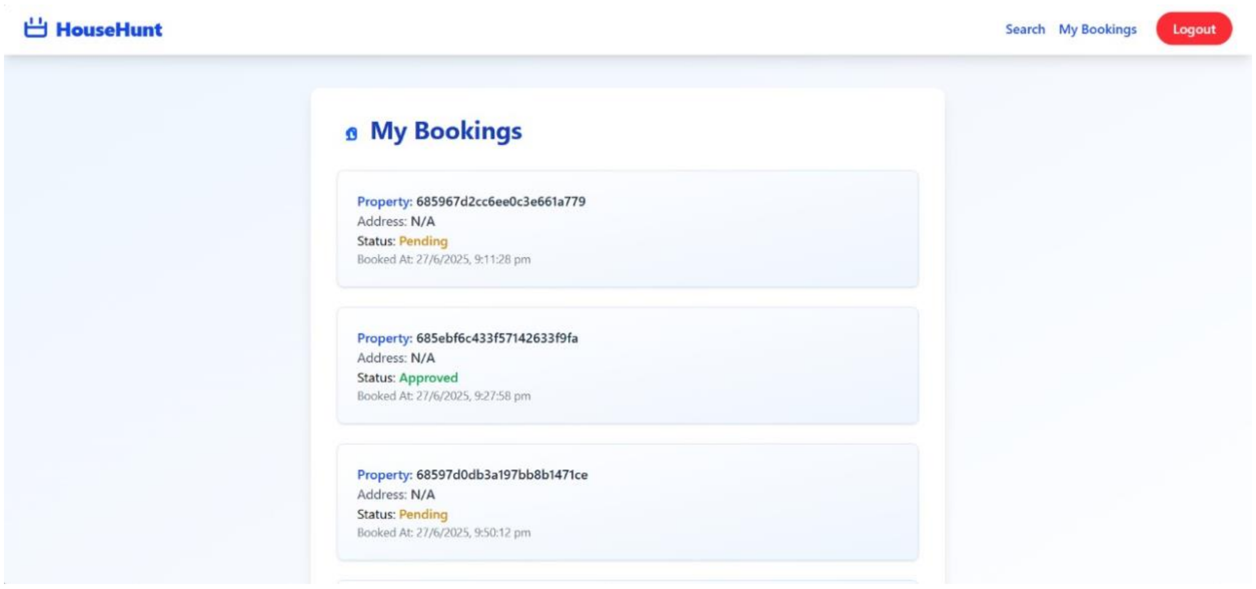
- **Property detail view with inquiry button:** A detailed look at an individual property's page, including images, description, amenities, and the prominent "Inquire Now" button.



Manage Properties

Apartment | Rent
bhimavaram
1234567890 | 15000
a,a
apartment in byrm

- **Owner dashboard with "Add Property" option:** A depiction of the owner's management interface, featuring options to add new listings and manage existing ones.



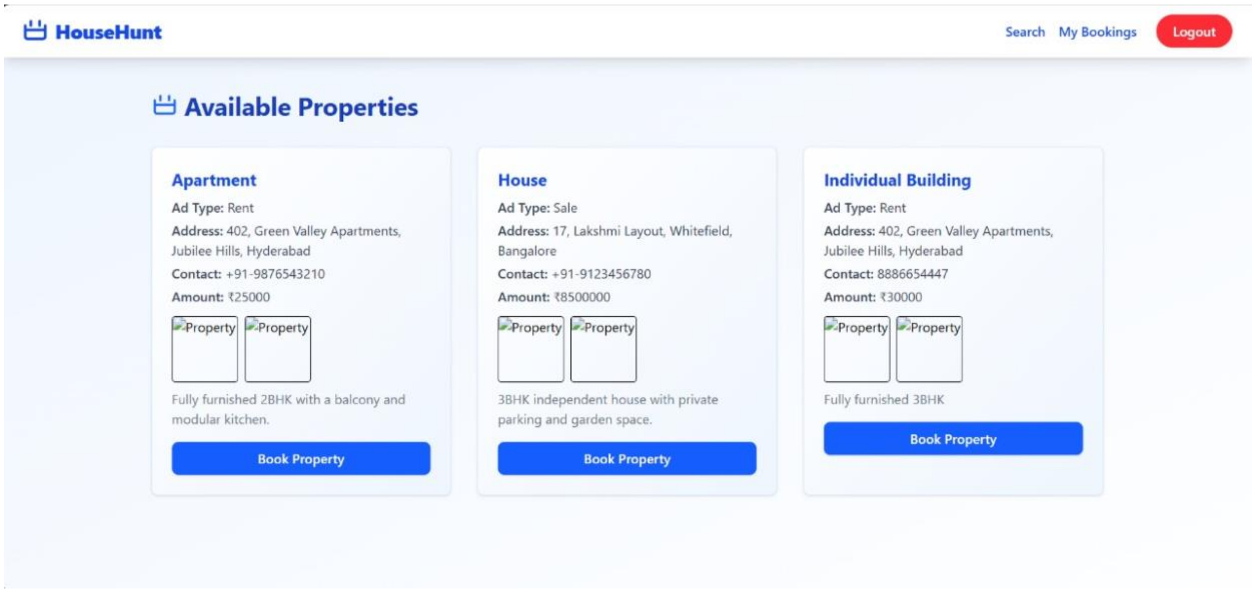
My Bookings

Property: 685967d2cc6ee0c3e661a779
Address: N/A
Status: Pending
Booked At: 27/6/2025, 9:11:28 pm

Property: 685ebf6c433f57142633f9fa
Address: N/A
Status: Approved
Booked At: 27/6/2025, 9:27:58 pm

Property: 68597d0db3a197bb8b1471ce
Address: N/A
Status: Pending
Booked At: 27/6/2025, 9:50:12 pm

- **Admin panel with list of pending approvals:** A screenshot of the administrator's view, showing a list of properties or users awaiting approval.


HouseHunt

Search My Bookings Logout

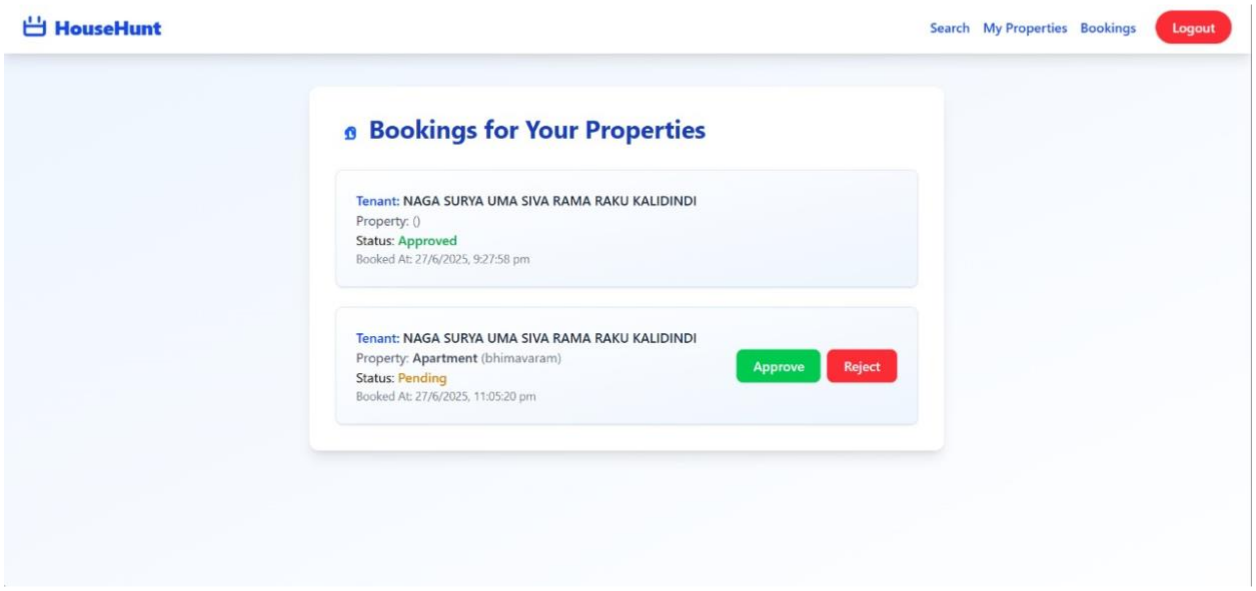
Available Properties

Apartment
Ad Type: Rent
Address: 402, Green Valley Apartments, Jubilee Hills, Hyderabad
Contact: +91-9876543210
Amount: ₹25000
Fully furnished 2BHK with a balcony and modular kitchen.
Book Property

House
Ad Type: Sale
Address: 17, Lakshmi Layout, Whitefield, Bangalore
Contact: +91-9123456780
Amount: ₹8500000
3BHK independent house with private parking and garden space.
Book Property

Individual Building
Ad Type: Rent
Address: 402, Green Valley Apartments, Jubilee Hills, Hyderabad
Contact: 8886654447
Amount: ₹30000
Fully furnished 3BHK
Book Property

- **Booking confirmation flow chart:** A visual representation of the steps involved in a successful booking process, from inquiry to confirmation.


HouseHunt

Search My Properties Bookings Logout

Bookings for Your Properties

Tenant: NAGA SURYA UMA SIVA RAMA RAKU KALIDINDI
Property: ()
Status: Approved
Booked At: 27/6/2025, 9:27:58 pm

Tenant: NAGA SURYA UMA SIVA RAMA RAKU KALIDINDI
Property: Apartment (bhimavaram)
Status: Pending
Booked At: 27/6/2025, 11:05:20 pm
Approve Reject

11.2. Visuals Creation

These visuals will be created and subsequently added to this documentation using Napkin to offer a comprehensive visual guide to the HouseHunt application.

---

12. Known Issues

As with any software project, there are identified areas for improvement and known limitations.

12.1. Mobile Responsiveness Needs Improvement

While the application is functional on mobile devices, the overall **mobile responsiveness requires further refinement** to provide an optimal user experience across various screen sizes and orientations.

12.2. Admin Panel Currently Lacks Pagination

The current **Admin panel does not yet implement pagination** for long lists of items (e.g., users, properties). This can make navigation cumbersome with a large dataset and will be addressed in future updates.

---

13. Future Enhancements

The HouseHunt project has a roadmap for continuous improvement and new feature integration.

13.1. Integrate Payment Gateway for Lease Transactions

A key future enhancement is the **integration of a robust payment gateway**. This will enable secure and direct financial transactions for lease agreements directly within the application, streamlining the rental process.

13.2. Enable In-App Notifications

Implementing **in-app notifications** will significantly enhance user engagement by providing real-time alerts for new inquiries, booking status updates, messages, and other important events.

13.3. Add Real-Time Chat Between Renter and Owner

Developing a **real-time chat feature** will facilitate instant communication between renters and property owners, making negotiations and clarifications more efficient and direct.

13.4. Implement Google Maps API for Location View

Integrating the **Google Maps API** will allow renters to view property locations directly on a map, providing a visual context and aiding in neighborhood exploration.
