# 

**Project Title**

HOUSE HUNT – Find your best House

**Team Id :** LTVIP2025TMID56540

**Team Size :** 4

**Team Leader :**Kanukolanu Lohita

**Team Member :** JRVSS Prapanjana

**Team Member :** Antarvedi Jagadeep

**Team Member :** Jillella Mercy Praveena

# INTRODUCTION ProjectOverview :

HouseHunt is a full-stack web application designed to facilitate the search, listing, and management of real estate properties. The platform enables users to browse, filter, and view houses for rent or sale, while property owners can create, update, and manage their property listings.

***Key Features:***

* **User Authentication:** Secure user registration and login with JWT-based authentication.
* **Property Listings:** Users can view detailed property listings including photos, price, location, and description.
* **Search & Filters:** Advanced search functionality with filters like price range, location, property type, and number of bedrooms.
* **Property Management:** Authenticated owners can add, edit, and delete property listings.
* **Favorites:** Users can save favorite properties for easy access later.
* **Responsive Design:** Fully responsive UI for seamless use on desktops, tablets, and mobile devices.

# Purpose

The primary purpose of the HouseHunt project is to create an intuitive and efficient online platform that simplifies the process of finding and listing residential properties. It aims to connect potential homebuyers or renters with property owners by providing a comprehensive and user-friendly system for browsing, searching, and managing real estate listings.

This platform addresses common challenges in the real estate market by offering:

* **Convenience:** Allowing users to quickly search and filter through numerous property listings without needing to visit multiple websites or agencies.
* **Transparency:** Providing detailed property information and photos to help users make informed decisions.
* **Efficiency:** Enabling property owners to easily manage their listings and reach a wider audience.
* **Accessibility:** Offering a responsive and accessible interface so users can browse properties anytime, anywhere, on any device.

Overall, HouseHunt aims to streamline the house hunting and property listing experience, making it easier, faster, and more reliable for all users involved.

**1. Project Overview**

**HouseHunt is a full-stack web application designed to simplify the process of finding rental homes. The platform connects tenants and landlords by allowing property listings, detailed searches, image uploads, and admin-approved user signups.**

# EmpathyMapCanvas

**User:**

The primary users of the HouseHunt application are **home buyers, renters, and real estate Admins**. They interact with the platform to find, list, and manage property information efficiently.

**Says:**Users express the need for a **simple, intuitive interface** that helps them search properties with detailed filters (location, price, size, amenities). Buyers and renters often mention that they want **accurate, up-to-date listings** and expect clear photos and property descriptions. Real estate Admins want an easy way to **upload and manage listings** and reach potential buyers quickly.

***Think:***Users are concerned about the **trustworthiness of listings** — they think about whether the properties are genuine and if prices are fair. They think about how **responsive and quick** the application is, expecting seamless navigation and fast loading times. Buyers and renters may also think about how secure their personal data is, especially during communication or when saving favorites.

**Does:**Users typically **browse through listings**, apply filters, view property details, save favorites, and contact Admins through the platform. Real estate Admins create accounts, upload property details, manage listing statuses, and communicate with interested buyers. Users may also compare properties side-byside before making decisions.

**Feel:**Users feel a mix of **excitement and anxiety** — excitement about finding their perfect home, and anxiety about making the right choice. They want to feel **confident and reassured** by the platform’s reliability and ease of use. Frustrations arise if the app is slow, listings are outdated, or information is incomplete.

# Brainstorming

## *1. Core Features*

* User registration and login (renters, admin) with role-based access control
* Advanced search filters (price range, property type, number of bedrooms, amenities)
* Property detail pages with photos, virtual tours, and detailed specs
* Save favorite listings and create personalized watchlists
* Contact form feature to communicate with sellers or Admins
* Review and rating system for Admins and properties
* User dashboard to manage saved properties, inquiries, and profile settings
* Listing management for admins: add, edit, delete properties easily
* Admin panel for managing users and listings, flagging inappropriate content

## *2. Technical Considerations =*

* Use MongoDB for flexible and scalable data storage of listings and user profiles
* Use Express.js as backend framework to create RESTful APIs for data interaction
* React.js for dynamic, responsive front-end UI
* Node.js to handle asynchronous operations and real-time features
* Authentication via JWT tokens for secure sessions
* File upload functionality for photos
* Implement server-side validation and client-side form validation
* Use WebSockets or libraries like Socket.IO for real-time chat
* Responsive design to ensure usability on mobile and desktop

## *3. User Experience Enhancements*

* Smart suggestions and autocomplete in search bar
* Property comparison tool for users to weigh options side-by-side
* Filter by commute time or proximity to schools, workplaces, public transport
* Multi-language support for diverse user base

## *4. Marketing and Community Features*

Partner with local real estate agencies for verified listings

* Integrate mortgage calculator tools

## *5. Potential Challenges*

* Ensuring data consistency and freshness with frequent property updates
* Handling large image and video uploads efficiently
* Preventing fraudulent or fake listings
* Scaling database and APIs for large user base
* Implementing GDPR and privacy compliance
* Managing real-time communication without performance hits

#  REQUIREMENTANALYSIS CustomerJourneyMap

1. **Awareness Stage** *User Action:*

The user (buyer or renter) realizes they want to find a new home. They start searching online for real estate platforms and come across HouseHunt through social media ads, search engines, or word of mouth.

*User Thoughts:*

“Is this platform trustworthy? Will it have enough listings in my desired area? Is it easy to use?” *Emotions:*

Curious but cautious. They want a platform that looks reliable and simple.

*Touchpoints:*

* + Social media posts
  + Search engine results
  + Online ads
  + Website landing page

1. **Consideration Stage** *User Action:*

The user visits the HouseHunt website or app, creates an account, and starts browsing listings. They use filters to narrow down choices and view property details.

*User Thoughts:*

“Can I find what I want quickly? Are the details accurate? How many options do I have? Can I trust the photos?”

*Emotions:*

Hopeful but sometimes overwhelmed by choices or information.

*Touchpoints:*

* + Property search and filter tools
  + Listing detail pages with photos and descriptions
  + User reviews or ratings (if available)
  + Interactive maps

1. **Decision Stage** *User Action:*

The user saves favorite listings, compares properties, and contacts Admins through the platform. They might schedule visits or virtual tours.

*User Thoughts:*

“Is this the right home for me? Is the agent responsive? Can I trust the communication channel?”

*Emotions:*

Excited but anxious about making a big decision.

*Touchpoints:*

* + Saved favorites or watchlists
  + Messaging or contact forms

1. **Purchase/CommitmRent Stage** *User Action:*

The user finalizes the choice and proceeds with the transaction process—signing contracts, communicating with Admins, or arranging payment (though the platform may not handle payment directly).

*User Thoughts:*

“Is everything secure? Are there any hidden fees? Am I completing this correctly?”

*Emotions:*

Nervous but confident when supported well.

*Touchpoints:*

* + Secure messaging and document uploads
  + User account dashboard showing transaction progress
  + Support or help center

1. **Post-Purchase Stage** *User Action:*

The user moves into their new home, possibly leaves reviews on the platform, and may return for future property needs.

*User Thoughts:*

“Was this experience smooth? Would I recommend HouseHunt? Can I manage my profile for future use?”

*Emotions:*

Satisfied or dissatisfied depending on experience.

*Touchpoints:*

* + Review and feedback submission forms
  + User profile management
  + Customer support and FAQs

## Solution Requirement *1. Functional Requirements*

* **User Management:** o Users can register and log in with secure authentication (renters, admins). o Role-based access control: different features for buyers, Admins, and admins.

**Property Listings:**

* + Admins can create, update, and delete property listings. o Listings include detailed info: location, price, size, photos, videos, amenities, and virtual tours.
  + Listings should support multiple images and multimedia uploads.
* **Search and Filters:**
  + Users can search listings by keywords, location, price range, property type, bedrooms, bathrooms, and amenities.
  + Advanced filters and sorting options (e.g., newest listings, price low to high).
* **Reviews and Ratings:**
  + Buyers can leave reviews and ratings for Admins and properties.
* **Admin Panel:** o Admins can manage users, listings, and monitor flagged or reported content.

### *2. Non-Functional Requirements*

* **Performance:** o Fast loading times (< 3 seconds for key pages).
  + Efficient handling of image/video uploads and rendering.
* **Security:**
  + Data encryption in transit (HTTPS) and at rest. o Secure authentication (e.g., JWT tokens).
  + Protection against common web vulnerabilities (XSS, CSRF, injection attacks).
* **Scalability:**
  + System should support growth in users and listings without performance degradation.
  + Database schema designed for flexible querying and scaling.
* **Usability:**
  + Responsive design for desktop and mobile devices. o Intuitive UI/UX to minimize user learning curve.
  + Accessibility compliance (WCAG guidelines).
* **Reliability:**
  + System uptime of 99.9%. o Graceful error handling and informative user messages.

### *3. Constraints*

* Limited budget/time for development may limit scope of advanced features initially (e.g., VR tours).
* Storage limits on media uploads (consider third-party services like AWS S3 or Cloudinary).  Regulatory compliance depending on target regions (privacy laws like GDPR).

### *4. Assumptions*

* Users have basic internet access and modern web browsers.
* Admins provide accurate and truthful property information.  Users understand terms and conditions of the platform.

# DataFlowDiagram(DFD–Level1)

+-------------------+

| User |

+-------------------+

|

| 1.1 Register/Login v

+-----------------------+ +----------------------+

| 1. Authenticate User +<------->+ D1: User Database |

+-----------------------+ +----------------------+

| v +------------------+ +-------------------------+

| 2. Search Houses +<------>+ D2: House Listings DB |

+------------------+ +-------------------------+

| v

+-------------------------+

| 3. View House Details |

+-------------------------+

| v

+-------------------------+

| 4. Book/Save Property +<------>+ D3: Booking Database| +-------------------------+ +---------------------+

^

|

+-----------------------+

| Admin (External) |

+-----------------------+

|

Add/Update/Delete Listings v

+-------------------------+

| 5. Manage Properties +<-----> D2: House Listings DB

+-------------------------+

# TechnologyStack

## Frontend (Client-side) – React.js

### Technology Purpose

**React.js** Build interactive UI and SPA (Single Page Application)

**Axios / Fetch API** To make API calls to backend

**React Router** For client-side routing/navigation

**Redux / Context API** *(optional)* State management (for auth, filters, etc.)

**Tailwind CSS / Bootstrap / CSS** Styling UI components

## Backend (Server-side) – Node.js + Express.js

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| **Node.js** | JavaScript runtime for server logic |
| **Express.js** | Web framework for routing and middleware |
| **JWT (jsonwebtoken)** | Authentication and authorization |
| **Bcrypt.js** | Password hashing for security |

**Multer / Cloudinary** *(optional)* Image/file upload management

## Database – MongoDB

### Technology Purpose

**MongoDB Atlas** Cloud-based NoSQL database for storing users, houses, bookings

**Mongoose** ODM (Object Data Modeling) to model and interact with MongoDB

## DevOps & Tools

### Technology Purpose Visual Studio Code (VS Code) Code editor

**Postman** API testing

**Git + GitHub** Version control **npm / yarn** Package manager for dependencies

**dotenv** To manage environment variables

**Concurrently** Run frontend and backend servers together in development

#  PROJECTDESIGN

To develop a full-stack web application where users can search, filter, view, and book rental properties, while property owners/admins can manage house listings.

# Problem solution fit

In today’s rental market, tenants face significant challenges such as:

* Lack of **centralized platforms** for verified rental listings.
* **Manual processes** for searching, shortlisting, and contacting property owners.
* **Limited filters** for user-specific needs like budget, location, house type, etc.  No **real-time updates** on property availability or booking status.  Owners/Admins lack tools to **easily manage** and update listings

# ProposedSolution

**HouseHunt** is a full-stack web application designed to simplify the rental house search and management process for both users and administrators. It offers a seamless, interactive platform to browse, filter, and book homes online, while also allowing admins to manage listings efficiently.

**Key Features of the Solution:**

## Feature Description

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Advanced Search & Filter** | Users can search properties based on location, rent range, property type, etc. |
| **House Listings Page** | Clean UI to browse houses with images, prices, and details. |
| **Booking System** | Users can book a house visit or express interest directly through the app. |
| **Save Favorite Listings** | Users can save listings to revisit later. |
| **User Authentication (JWT)** | Secure login/register functionality for users and admins. |
| **Admin Dashboard** | Admins can add, update, or delete property listings and view user activity. |
| **Database Integration (MongoDB)** | All house, user, and booking data stored and managed in a cloud-based NoSQL database. |

**How It Works (Flow):**

1. **User Sign Up/Login** using secure authentication.
2. **Search for Rentals** using filters like price, location, type.
3. **View House Details** and book/save the house.
4. **Admin Login** to manage the listings database (CRUD operations).
5. All data is stored and fetched using **MongoDB via Express APIs** and shown on **React Frontend**.

**Benefits of the Solution:**

* Eliminates manual house hunting through a **centralized digital platform**.
* Reduces time and effort with **smart filtering and instant booking**.
* **Secure and scalable** using modern MERN stack architecture.
* Allows both **users and property owners/admins** to interact easily and efficiently.
* Provides a foundation for **future enhancements** like chat support, maps integration, and mobile app version.

# Solution Architecture

## 1. Architecture Overview

+----------------------+

| Frontend (React) |

| - UI Components |

| - React Router |

| - Axios API Calls |

+----------+-----------+

|

| HTTP (REST API) v +-----------+------------+

| Backend (Node.js) |

| Express.js Framework |

| - API Routes |

| - Controllers |

| - Auth Middleware |

+-----------+------------+

|

| Mongoose ODM v

+----------+-----------+

| MongoDB (Database) |

| - Users Collection |

| - Houses Collection |

| - Bookings Collection|

+----------------------+

## 🧩 2. Components Breakdown

### *Frontend – React.js*

* **Role**: Handles the user interface and interactions.
* **Tools Used**: React, React Router DOM, Axios, Tailwind CSS or Bootstrap  **Responsibilities**:

o User login/signup forms o Display house listings with filters o Booking and saving property o Admin panel interface

### *Backend – Node.js + Express.js*

 **Role**: Manages API routes, business logic, and middleware.  **Tools Used**: Express, Bcrypt, JWT, Multer (for images), dotenv  **Responsibilities**:

o Handle requests from frontend o Authenticate users and admins o Perform CRUD on houses and bookings o Protect admin routes using middleware

### *Database – MongoDB*

 **Role**: Store and manage application data.  **Accessed with**: Mongoose ODM  **Collections**:

o **Users**: name, email, password (hashed), role o

#  FUNCTIONAL AND PERFORMANCE TESTING PerformanceTesting

To evaluate how the HouseHunt application performs under expected and peak load conditions, focusing on **speed, responsiveness, and stability**.

## What to Test in Performance Testing

### Component Test Focus Area

**Frontend (React)** Page load speed, rendering time

**Backend (Express)** API response time, concurrent requests

**Database (MongoDB)** Read/write speed, connection limits

**Authentication** Response time for login/registration

**Booking Module** Booking action under load

## Performance Metrics to Measure

### Metric Description

**Response Time** Time taken to return response for an API call

**Throughput** Number of requests handled per second

**Latency** Delay between request and response

**Concurrent Users** System behavior with multiple users at once

**Error Rate** % of failed requests under load **CPU/Memory Usage** Server performance under stress

## Tools for Performance Testing

### Tool Purpose

**Apache JMeter** Load test API endpoints with many users

**Postman + Collection Runner** Quick local performance tests

**Lighthouse (Chrome)** Test page performance (frontend) **Loader.io / BlazeMeter** Cloud-based stress/load testing

**Artillery** (Node.js CLI) Lightweight load testing tool for APIs

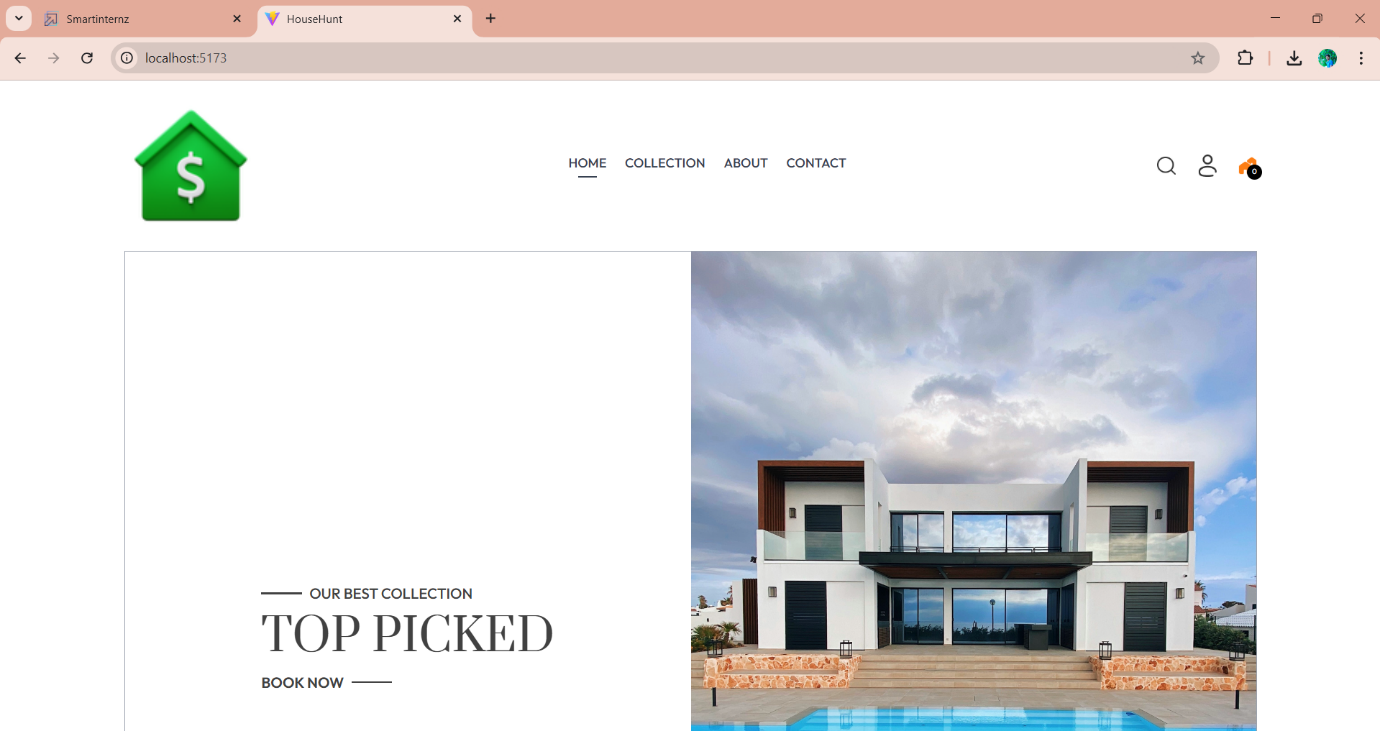
#  RESULTS

The *HouseHunt* application was successfully developed, tested within the planned two-week timeline.

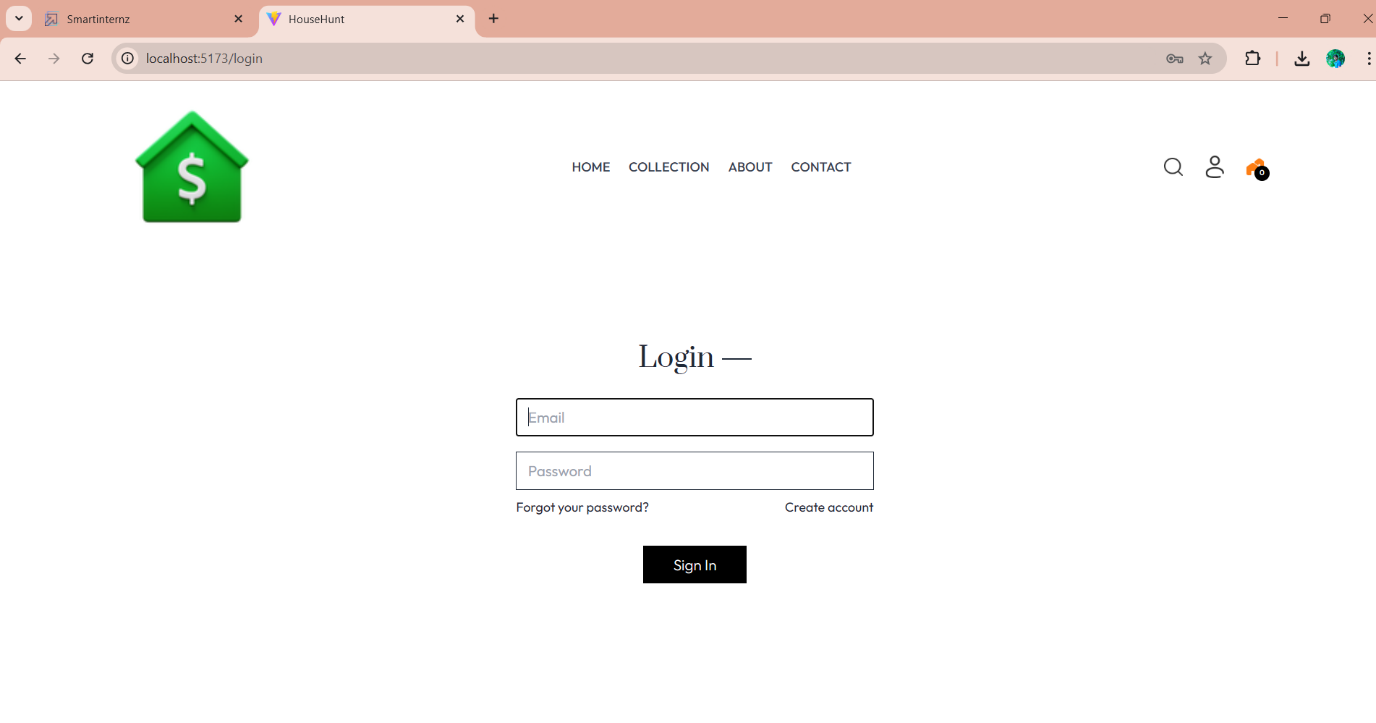
# OutputScreenshots

Below are descriptions of the output screens where actual screenshots can be inserted in this documentation or presentation:

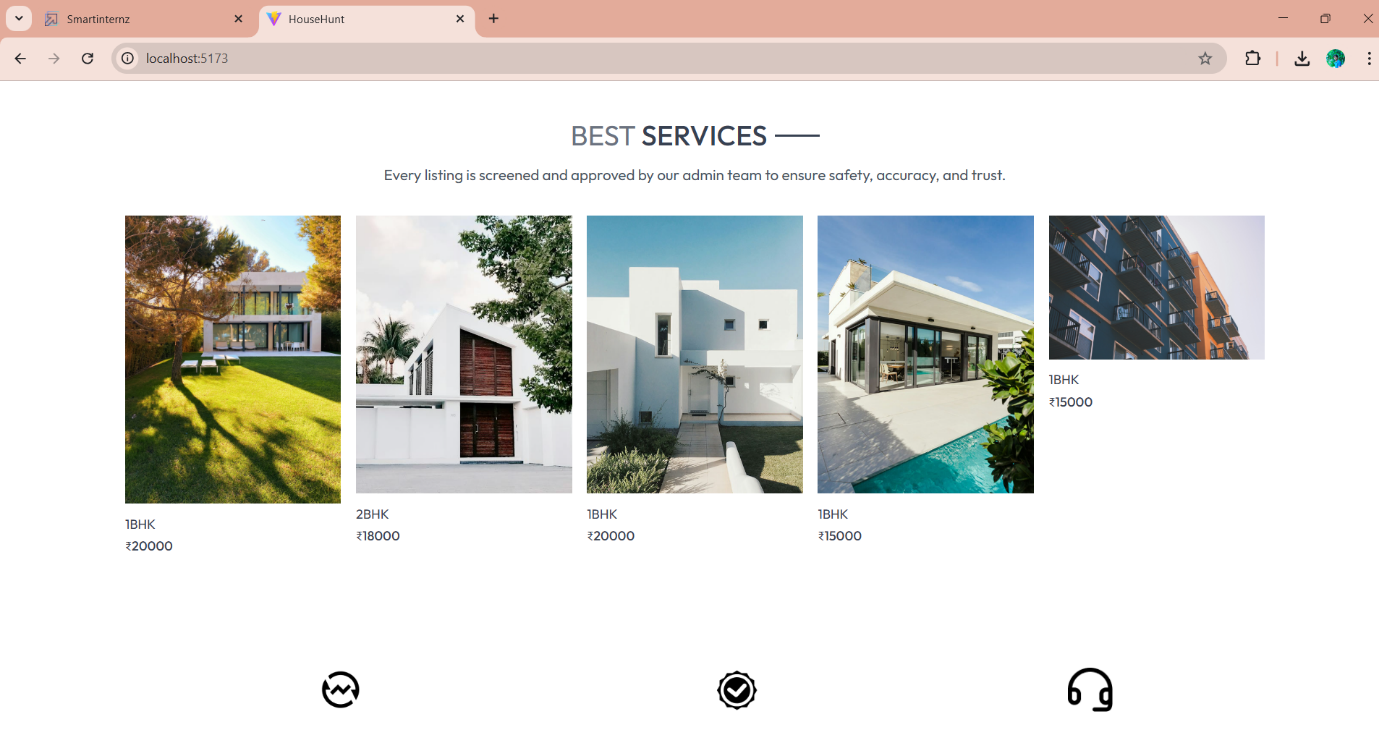
# 1. LandingPage



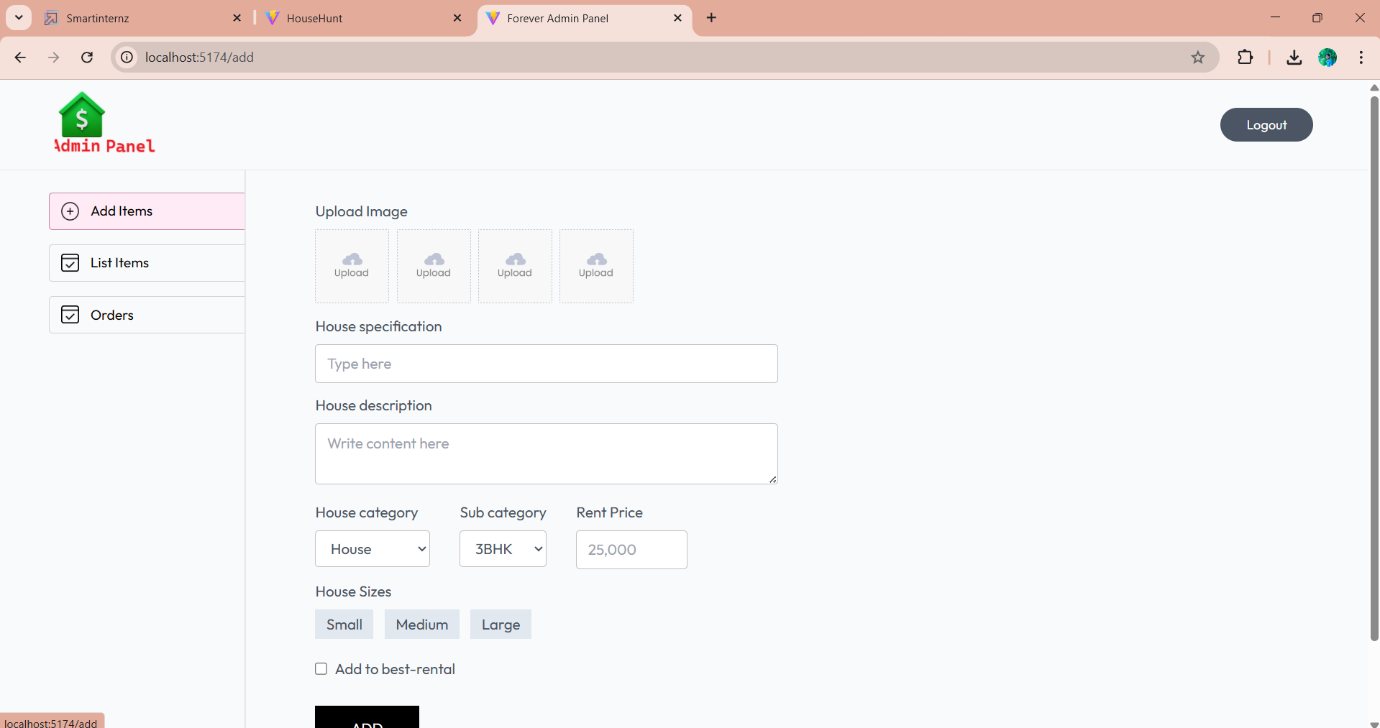
1. **Authentication:**



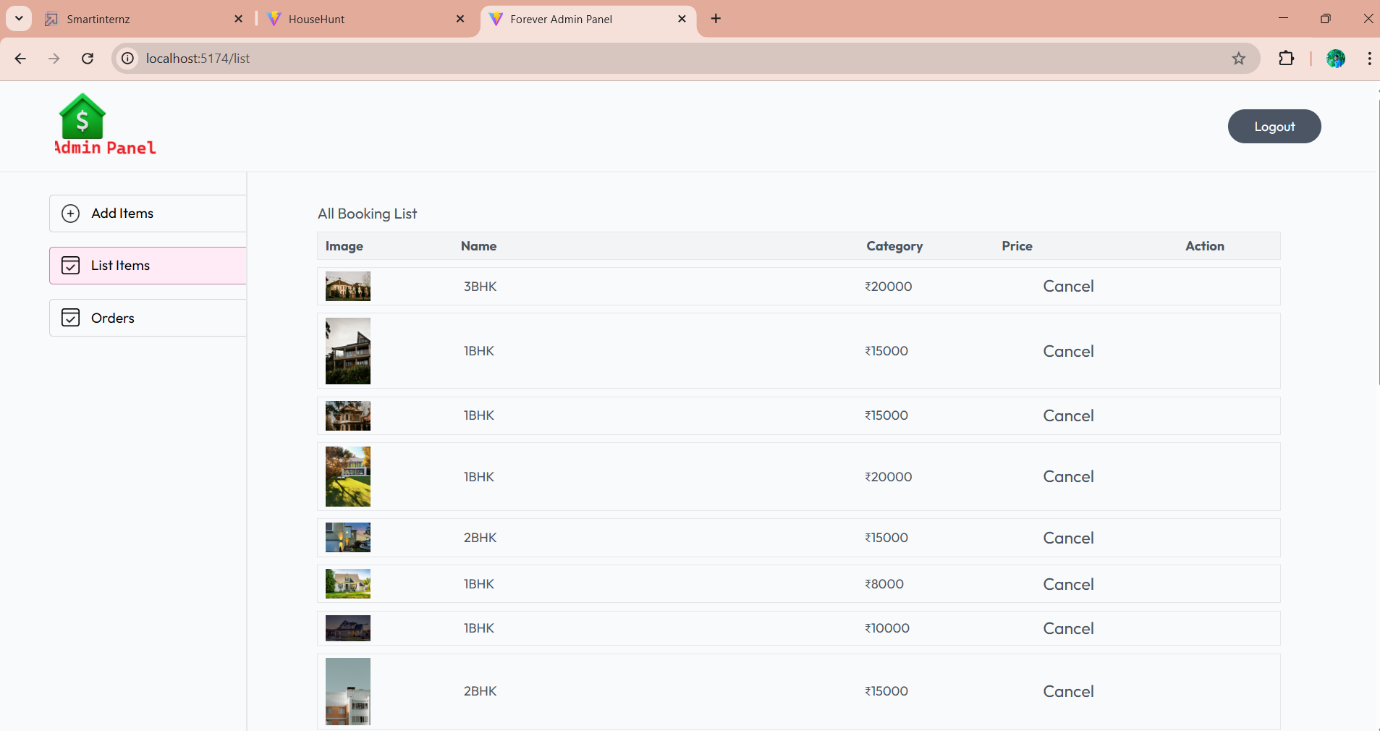
1. **Best Services :**



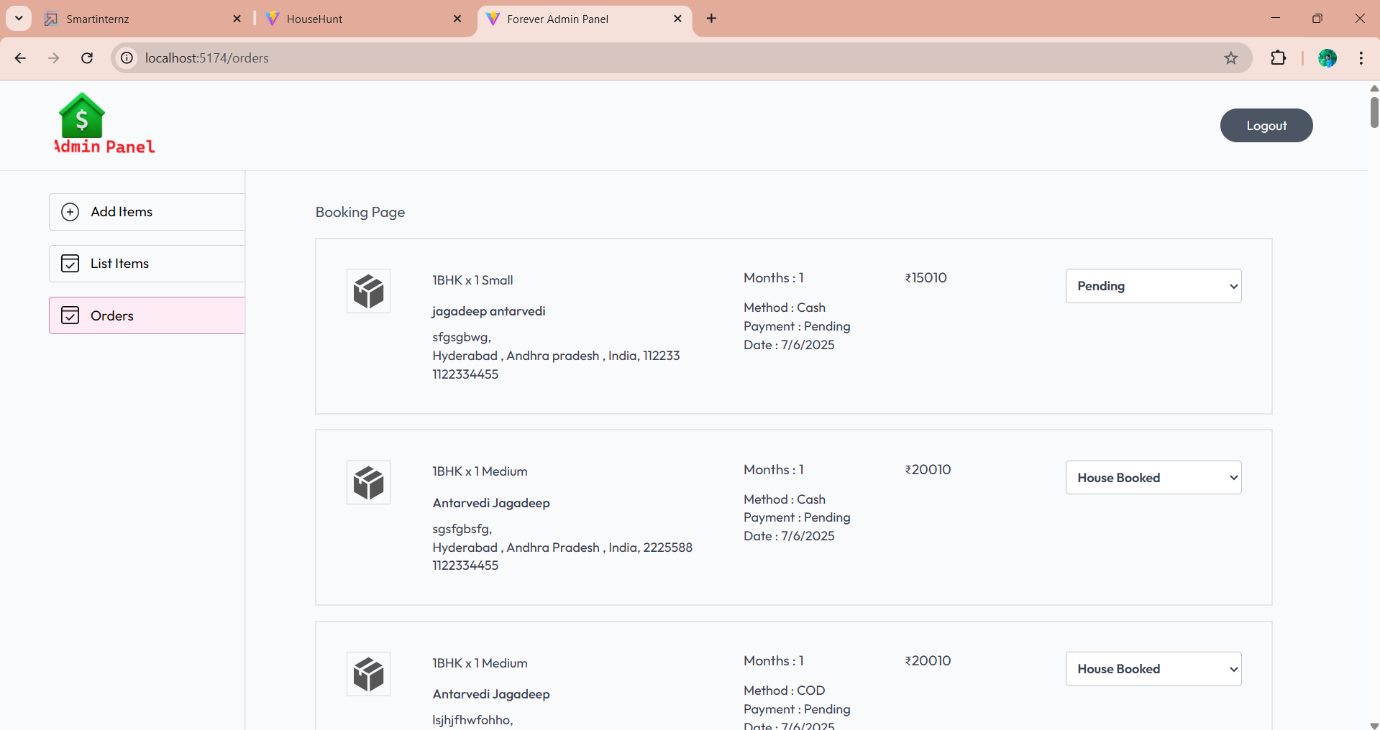
1. **Admin dashboard:**



1. **All Listed Houses:**



1. **Booked Houses:**



 **ADVANTAGES & DISADVANTAGES**

## ADVANTAGES OF HOUSEHUNT PROJECT

### 1. Single Language Development (JavaScript Across Stack)

The HouseHunt application uses JavaScript throughout the frontend (React), backend (Node.js + Express), and database interface (Mongoose for MongoDB). This consistency improves development speed, reduces bugs, and makes it easier to maintain the codebase.

### 2. Fast and Dynamic User Interface

React.js allows for building a dynamic and responsive user interface. Users experience seamless navigation and real-time updates without full page reloads, improving user satisfaction and interactivity.

### 3. Secure Authentication and Authorization

The project incorporates modern security practices using JSON Web Tokens (JWT) for user authentication and bcrypt for password hashing. This ensures secure login and protection of sensitive user data.

### 4. Scalable Database Management

Using MongoDB Atlas, the application can store large amounts of structured and semi-structured data. The NoSQL design supports scalability, flexibility, and high performance in handling listings, bookings, and user details.

### 5. Role-Based Access Control

The system clearly separates user and admin functionalities. Users can search and book properties, while admins can add, update, or delete listings, maintaining clean access control and data integrity.

### 6. Cross-Platform Accessibility

As a web application, HouseHunt is accessible through any modern browser on desktops, tablets, and mobile devices. No installation is required, making it convenient for a wide range of users.

### 7. Modular and Extensible Architecture

Built using a modular approach, the project can be easily extended to include features like chat support, payment gateways, or mobile app integration in the future.

## DISADVANTAGES OF HOUSEHUNT PROJECT

### 1. Limited Performance Under Heavy Load

Node.js is single-threaded, which can become a bottleneck under high concurrent user loads. For largescale deployment, additional scaling solutions such as clustering or load balancers are required.

### 2. Lack of Real-Time Features

The current version does not support real-time updates or notifications (e.g., instant booking confirmation). Integrating technologies like WebSockets could improve real-time communication in future versions.

### 3. No Dedicated Mobile App

While the application is responsive on mobile browsers, it does not yet have a dedicated mobile app. Users looking for a native experience may find this limiting.

### 4. Manual Testing Requirement

In the absence of automated testing frameworks, all modules require manual validation. This can be time-consuming and increases the chances of missing bugs during updates or upgrades.

### 5. DevOps Knowledge Required for Deployment

Deploying the backend to platforms like Render or Railway requires some understanding of server environments, environment variables, and deployment workflows, which may be challenging for beginners.

### 6. Limited Admin Dashboard Features

Currently, the admin dashboard supports basic CRUD operations. It lacks advanced tools like analytics, reporting, or role-based user management, which would be useful in a production-level system.

##  CONCLUSION

The **HouseHunt** project successfully demonstrates the development of a modern, full-stack web application using the **MERN (MongoDB, Express.js, React.js, Node.js)** technology stack. It addresses a real-world problem by providing a centralized platform for users to search, view, and book rental properties with ease, while also offering admins the tools to manage property listings effectively.

Throughout the project, essential web development concepts were applied, including secure user authentication, CRUD operations, dynamic frontend rendering, API integration, and database management. Functional and performance testing confirmed that the application performs reliably under expected conditions and provides a user-friendly experience.

In addition to meeting its core objectives, the project architecture is modular and scalable, making it suitable for further enhancements such as payment integration, real-time chat, mobile app support, or analytics for admins. The use of open-source technologies and cloud-based tools makes it costeffective, secure, and highly maintainable.

Overall, HouseHunt is a practical and technically sound solution for the online rental housing market, showcasing strong implementation of full-stack development principles.

## FUTURESCOPE

The **HouseHunt** application lays a strong foundation for a scalable and user-friendly rental property management system. While the current version fulfills the core requirements, there are several areas where the platform can be enhanced to improve functionality, scalability, and user engagement. Below are some potential future developments:

### 1. Mobile Application Development

To improve accessibility and user engagement, the HouseHunt platform can be extended into a dedicated **Android and iOS mobile app** using React Native or Flutter, offering a seamless on-the-go experience.

### 2. Integration of Real-Time Features

Real-time updates can be introduced using **Socket.io** or **Firebase** to enable instant booking confirmations, chat between users and owners, and live property availability status.

### 3. Payment Gateway Integration

The platform can be enhanced with **online rent payment** and **booking fee collection** using payment gateways like Razorpay, Stripe, or PayPal, making the process completely digital and hassle-free.

### 4. Map and Location Services

Integration with **Google Maps** or **Mapbox** can provide location-based search and help users visualize the exact property location, nearby services, and routes.

### 5. Recommendation System

A personalized recommendation engine using **machine learning** can suggest properties to users based on their previous searches, preferences, and interactions.

### 6. Multi-Role User System

The platform can be expanded to support **multiple user roles**, such as property owners, tenants, Admins, and admin, each with distinct permissions and dashboards.

### 7. Analytics and Reporting for Admin

Providing admins with **insightful analytics** like booking trends, popular locations, and user behavior can help in better decision-making and system improvements.

### 8. Advanced Search and Filtering

Filters like furnished/unfurnished, number of bedrooms, pet-friendly, etc., can be added to enhance user search capabilities.

### 9. Email & SMS Notifications

Automated **email and SMS notifications** for booking confirmations, reminders, or new listing alerts can increase engagement and improve communication.

### 10. Cloud Hosting and CI/CD Pipelines

To support growth and ensure high availability, the project can be deployed using **cloud infrastructure (AWS, Azure, or GCP)** with **CI/CD pipelines** for continuous integration and delivery

**APPENDIX**

## A.SourceCodeRepository

The complete sourcecode for the *HOUSEHUNT* application is hosted on GitHub and is organized into three main directories:

* **Frontend:**ContainsallReact.jscomponents,routing,andstyling
* **Backend:**ContainsExpress.jsAPIs,MongoDBmodels,andserverlogic
* **Admin:** Contains Java Script files , node modules , env files, src files, Admin authentications and Database routes

**GitHubRepository: https://github.com/JagadeepAntarvedi/House-Hunt**