**Full Stack Development with MERN – Project**

**FSD DOCUMENTATION**

**Project Title**:

MERN House Rent App - HouseHunt

**Team Members**:

Yash Tilwani (22BCE10405)

Anshika Mishra (22BCE10635)

Viraj Patidar (22BCE10531)

Anukul Tripathi (22BAI10145)

1. **Introduction**

In an era where mobility is integral to our lives, finding rental properties has become increasingly challenging. This holds especially true for students, professionals, and families who often face a maze of options and unreliable methods when moving to unfamiliar cities. Conventional approaches to house hunting, such as relying on brokers, verbal referrals, or navigating through countless social media posts, often prove to be time-consuming and frustrating. These methods frequently lack transparency, leading to misinformation and unnecessary delays in securing a rental property.

To address these challenges, **HOUSE HUNT**, a modern web-based application built on the MERN stack (MongoDB, Express.js, React.js, and Node.js), emerges as a solution aimed at redefining the house rental process. Leveraging cutting-edge technology, HOUSE HUNT offers a seamless experience for both landlords and tenants. Landlords can list properties with verified details, ensuring credibility and reliability. Tenants, on the other hand, gain access to a streamlined search feature that allows them to explore homes based on their preferences, including location, budget, and required amenities.

The application goes beyond basic functionalities by integrating advanced features that enhance convenience and security. These include user authentication to protect personal information, secure channels for communication between landlords and tenants, and real-time updates that foster transparency and trust. By bridging the gap between landlords and tenants, HOUSE HUNT eliminates common barriers in the rental process, such as miscommunication or delays, creating a unified digital platform.

The primary goal of HOUSE HUNT is to deliver convenience, trust, and efficiency in one comprehensive solution. This project aspires to redefine how people approach rental property searches, empowering users with accurate information and cutting-edge tools. By combining innovative design with practical usability, HOUSE HUNT promises to transform house hunting into a hassle-free and rewarding experience for all.

1. **Project Overview**

HouseHunt is a cutting-edge, full-stack web application designed to revolutionize the way individuals find, book, and manage properties online. Inspired by the ease and convenience of platforms like Airbnb, HouseHunt takes it a step further by offering a sleek, intuitive interface and a seamless user experience. The platform is built to cater to both property owners looking to list their spaces and renters searching for their next home, whether it's for a weekend getaway or a long-term stay.

**Key Aspects of HouseHunt:**

**1. User-Friendly Interface:**

HouseHunt is designed with a focus on simplicity and accessibility. Whether you're tech-savvy or a first-time user, the platform ensures that navigating through property listings and booking a stay is as easy as possible. The clean design and responsive layout make the platform equally functional on both desktop and mobile devices. We've employed **Material-UI (MUI)** to create a consistent and professional design, which is optimized for performance and aesthetics.

**2. Intuitive Property Search and Filters:**

One of the standout features of HouseHunt is its robust search functionality. Users can filter properties based on various criteria, such as:

* **Location**: Search by city, neighborhood, or even specific landmarks.
* **Price Range**: Whether you're looking for a budget-friendly apartment or a luxury villa, our dynamic price filter helps you find options within your budget.
* **Property Type**: From cozy apartments to luxurious penthouses or beachfront cottages, HouseHunt supports a wide variety of property types.
* **Availability**: Easily check whether a property is available for the desired dates, with real-time availability updates.

Additionally, users can sort results by factors like price, user ratings, and proximity to certain attractions, making it even easier to find the perfect place that fits their needs.

**3. Seamless Booking Experience:**

HouseHunt makes the booking process a breeze. Once you’ve found the ideal property, the platform walks you through a smooth booking process. Users can select their check-in and check-out dates, input special requests, and confirm their reservation within just a few clicks. The process is quick, user-friendly, and designed to eliminate any unnecessary friction.

Once a booking is made, both the renter and the property owner receive notifications, and renters can view booking details directly from their profile, including the option to cancel or modify their reservation if needed.

**4. Detailed Property Information:**

Unlike many other platforms, HouseHunt places a heavy emphasis on providing users with comprehensive, up-to-date information about each property listing. Users can expect to find:

* **High-Quality Images**: A gallery of professional images showcasing the property’s features, rooms, and surrounding areas.
* **Property Descriptions**: Detailed descriptions that highlight the property’s unique features, including amenities, room layouts, and neighborhood details.
* **Availability Calendar**: A clear calendar that shows exactly when the property is available, allowing users to plan their stays accordingly.
* **Reviews and Ratings**: Honest feedback from previous renters, helping users make informed decisions based on others' experiences.
* **Owner Information**: Contact details for the property owner, including an option to message them directly via the platform for any queries or clarifications.

**5. Property Owner’s Dashboard:**

HouseHunt isn’t just for renters—it also serves as a powerful tool for property owners and managers. The owner’s dashboard provides a comprehensive overview of their listings, bookings, and financial transactions. Property owners can:

* **List New Properties**: Easily upload details about new properties, add descriptions, images, and set availability dates.
* **Manage Bookings**: View, accept, or decline booking requests, and keep track of past and upcoming reservations.
* **Monitor Earnings**: Access detailed reports about booking earnings, payments received, and upcoming payouts.
* **Communicate with Renters**: Send messages to renters, resolve any issues, or provide additional information about the property.

This powerful backend system ensures that property owners can manage their listings efficiently, saving time and enhancing their overall experience.

**6. Secure and Transparent Payment System:**

HouseHunt integrates a secure, third-party payment gateway that ensures all transactions are handled safely. Renters can make payments for bookings directly on the platform, using popular payment methods such as **credit cards, debit cards**, and **online wallets** like **PayPal**. Property owners can expect timely payments for bookings, while renters are assured of their funds being handled securely.

The payment system is fully transparent, offering both users and owners clear transaction histories and billing details, ensuring peace of mind for all parties involved.

**7. Real-Time Notifications:**

To keep users updated, HouseHunt employs a robust notification system. Whether it's confirming a new booking, notifying a user about a change in booking details, or informing an owner about an inquiry, notifications are delivered in real-time. Users receive timely alerts via **email**, **SMS**, or **in-app notifications**, keeping them engaged and up-to-date every step of the way.

**8. Scalable and Secure Platform:**

Built with the **MERN stack** (MongoDB, Express, React, Node.js), HouseHunt is a scalable, high-performance platform capable of handling large volumes of data and users. We prioritize security, with strong encryption practices to protect sensitive user data and transactions. **JWT (JSON Web Tokens)** ensure that only authenticated users can access sensitive areas of the platform, including booking management and property creation.

Additionally, the application is designed to scale easily as the user base grows, making it adaptable for the future. Whether you're running a small property rental business or managing hundreds of listings, HouseHunt is designed to accommodate your needs.

**9. Global Reach:**

Though initially focusing on a single region, HouseHunt has been built with global scalability in mind. In the future, we plan to expand our listings to cover a wide variety of locations worldwide, offering a truly global platform for both renters and property owners. Whether you're looking for a cozy apartment in the heart of Paris, a villa in the Mediterranean, or a remote cabin in the mountains, HouseHunt is the platform to help you find your perfect getaway.

**10. Future Growth and Enhancements:**

As the platform continues to grow, we plan on adding more features to enhance the user experience. Future plans include:

* **Smart Recommendation System**: Based on your preferences, past bookings, and searches, the platform will suggest properties that fit your style and budget.
* **Multilingual Support**: To accommodate users from all around the world, we will introduce multiple languages, ensuring a global reach.
* **Augmented Reality (AR) Property Tours**: Imagine being able to take a virtual tour of a property from the comfort of your home, using your phone or computer. We are looking into incorporating AR technology to bring this feature to life.

**Purpose:**  
The purpose of HouseHunt is to provide a robust and secure platform that allows users to browse through a variety of property listings, book their stays, and manage their profiles and bookings with ease. The goal is to make the entire process, from listing to booking, seamless for both users and property owners.

Additionally, this project was built as part of the **Smart Bridge Initiative**, aiming to develop real-world applications using modern technologies such as the **MERN stack**. This initiative focuses on solving practical problems while allowing developers to learn and grow their skills in full-stack web development.

**Features:**  
HouseHunt boasts a wide array of features that make property management and booking as easy as 1-2-3:

* + **Property Listings**: Users can view a variety of properties with detailed descriptions, images, prices, and availability.
  + **User Authentication**: Secure login and signup using JWT-based authentication.
  + **User Profile Management**: Users can manage their profiles and track their bookings.
  + **Booking System**: A seamless booking experience where users can reserve properties for a specific period.
  + **Admin Dashboard**: Admins can manage property listings, view bookings, and update property details.

The UI/UX is designed with simplicity in mind, using **Material-UI (MUI)** to ensure that users enjoy a clean, professional, and intuitive experience across all devices.

1. **Architecture**

**Frontend:**

The frontend of HouseHunt is built with **React 18**, using its component-based architecture to ensure a modular and scalable application. This structure makes it easy to add new features and components as the app grows.

For styling and responsive design, we rely on **Material-UI (MUI)**, which provides pre-built React components like buttons, sliders, and modals to quickly create an attractive and consistent UI across all devices.

**Redux Toolkit** is used for state management, allowing us to manage complex data flows, such as user authentication, property details, and bookings, in a predictable way. **Redux DevTools** help track state changes and debug the application efficiently.

**React Router** enables smooth, page-less navigation between different views, such as property listings, user profiles, and booking details, ensuring a seamless user experience on both desktop and mobile devices.

To interact with the backend, the frontend uses **Axios** to make GET, POST, PUT, and DELETE requests, ensuring real-time data updates and smooth user interactions.

**Key Frontend Technologies:**

* **React 18**: For building user interfaces with a component-based architecture.
* **Material-UI (MUI)**: For pre-built components and responsive design.
* **Redux Toolkit**: For managing global state and simplifying state management.
* **React Router**: For managing in-app routing and navigation.
* **Axios**: For making HTTP requests and handling API interactions.

**Backend:**

The backend of HouseHunt is built using **Node.js** and **Express.js**, which provide a lightweight and efficient framework for creating web applications. Node.js uses an asynchronous, event-driven architecture, ideal for handling multiple simultaneous requests without blocking operations. **Express.js** is used to build a RESTful API that handles user registration, property listings, and bookings.

The API follows **RESTful principles**, making it scalable and easy to add new features in the future, such as messaging or advanced property filters. For **user authentication**, we use **JWT (JSON Web Tokens)**. When users log in or register, they receive a token stored in **localStorage** and included in the **Authorization** header of subsequent API requests. This ensures that only authenticated users can perform actions like creating listings, making bookings, or viewing profiles.

Backend routes are secured using middleware that checks for a valid JWT before granting access to protected endpoints, such as property management or booking operations. If the token is missing or expired, the server denies access and prompts the user to log in again.

**Key Backend Technologies:**

* **Node.js**: For the server-side environment.
* **Express.js**: For building the RESTful API and handling routing.
* **JWT (JSON Web Tokens)**: For user authentication and authorization.
* **Middleware**: For securing routes and managing authentication flows.

**Database:**

**MongoDB** serves as the primary database for HouseHunt. We use **Mongoose**, an Object Data Modeling (ODM) library for MongoDB, to interact with the database. Mongoose simplifies database operations by providing a schema-based solution to model application data. This allows us to define the structure of the data, including validation and default values, while still taking advantage of MongoDB’s flexible document-based nature.

The database schema is designed to manage the core entities of the platform: **User**, **Property**, and **Booking**.

1. **User Schema**:
   * Contains information like the user’s name, email, password (hashed for security), role (e.g., renter, property owner), and booking history.
   * The password is encrypted using **bcrypt** before being stored in the database, ensuring that sensitive data is protected.
   * We also store an array of **bookings** related to the user, allowing them to track past and future reservations.
2. **Property Schema**:
   * Stores information about each property listing, such as **name**, **description**, **price per night**, **location**, **images**, and **availability**.
   * Each property can have multiple **images**, so we store URLs of images uploaded via an external cloud storage service (e.g., **AWS S3** or **Cloudinary**).
   * We also store **reviews** and **ratings** associated with each property, providing valuable feedback from renters that future users can view.
3. **Booking Schema**:
   * Manages the reservation details made by users, including **check-in and check-out dates**, **property ID** (to link the booking to a specific property), and **user ID** (to track which user made the booking).
   * Each booking also includes information about **payment status**, whether the booking was **confirmed** or **canceled**, and any **special requests** made by the user during booking.

**Key Database Technologies:**

* **ongoDB**: For storing and managing data with flexibility and scalability.
* **Mongoose**: For schema-based modeling and easy interaction with MongoDB.
* **bcrypt**: For securely hashing passwords before storing them.
* **Cloud Storage (e.g., AWS S3, Cloudinary)**: For storing property images and media files securely.

**Overall System Flow:**

1. **User Interactions**: When a user interacts with the frontend (e.g., searching for properties, making a booking), the frontend makes API calls to the backend to retrieve data or perform actions (like creating a booking).
2. **Backend Processing**: The backend processes these requests, performing the necessary logic (such as checking availability, validating the user's token, or saving the booking) and then returns the appropriate response to the frontend.
3. **Database Operations**: If the request involves data storage (e.g., creating a new property listing or storing a booking), the backend interacts with MongoDB to save the relevant data via Mongoose models.

This architecture ensures that HouseHunt operates efficiently, securely, and with the ability to scale as more users and features are added over time.

1. **Setup Instructions**

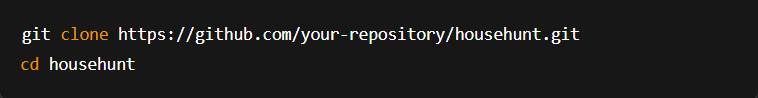
**Prerequisites:**

Before getting started, make sure you have the following installed on your system:

* **Node.js**: A JavaScript runtime used for the backend (version 14.x or later).
* **MongoDB**: A NoSQL database used to store application data.
* **npm**: A package manager for managing project dependencies.
* **Git**: For version control and project cloning.
* **Cloud Storage** (optional): If you’re handling image uploads, you may use services like AWS S3 or Cloudinary.

**Installation:**

**1.Clone the Repository**: First, clone the project repository to your local machine:



**2.Frontend Setup:**

* Navigate to the client directory and install the dependencies:



* Create a .env file in the client directory to store environment variables (e.g., API URLs).

**3.Backend Setup:**

* Navigate to the server directory and install the dependencies:



* Create a .env file in the server directory to store sensitive variables like database URLs, JWT secret, etc.

**4.MongoDB Setup:**

* If you're using a local MongoDB instance, make sure MongoDB is running on your machine.
* For production, use MongoDB Atlas to set up a cloud database.

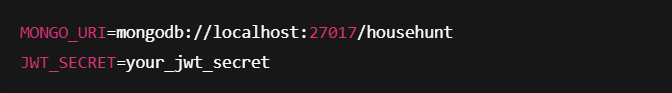
**5.Running the Application:**

To run the application locally, you’ll need to start both the frontend and backend servers:

* **Frontend**: Navigate to the client directory and run:



* **Backend**: Navigate to the server directory and run:



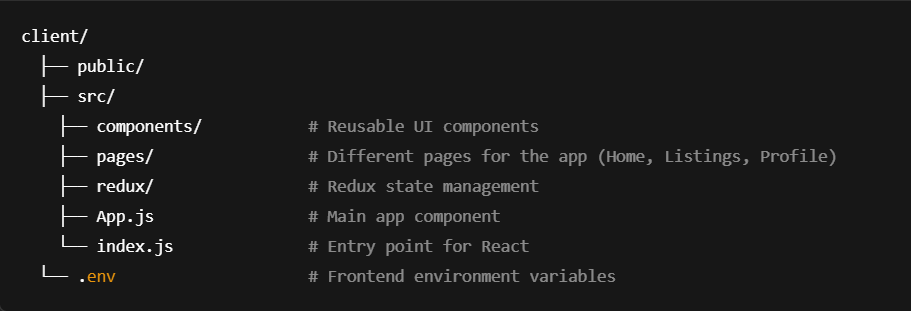
The app should now be running at http://localhost:3000 (or your specified port).

1. **Folder Structure**

**Client:**

The frontend is located in the client directory, and its structure follows a common React project setup:

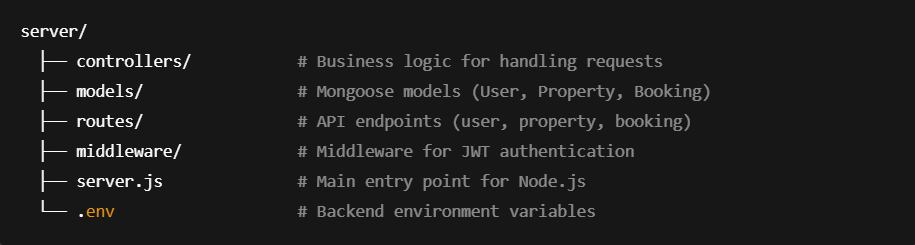
* **src/components**: Contains reusable React components (e.g., Header, PropertyCard, etc.).
* **src/pages**: Contains the different views or pages (e.g., HomePage, PropertyDetail, UserProfile).
* **src/redux**: Contains Redux slices and store configuration for state management.
* **src/api**: Contains files for handling API requests (e.g., Axios instance).
* **src/assets**: Contains images, styles, and other static assets.
* **src/utils**: Utility functions or constants used throughout the app.



**Server:**

The backend is located in the server directory, which is organized as follows:

* **models**: Contains Mongoose models (e.g., User, Property, Booking).
* **routes**: Contains API route handlers (e.g., userRoutes, propertyRoutes).
* **controllers**: Logic for handling API requests.
* **middleware**: Contains middleware like JWT authentication.
* **config**: Configuration files for database connection and environment variables.
* **utils**: Utility functions, such as password encryption and token generation.



1. **Running the Application**

To run **HouseHunt** locally, follow the steps below for both the **frontend** and **backend** servers:

**1. Frontend Setup**

1. **Navigate to the client directory**:



1. **Install Dependencies**:



1. **Start the Frontend Development Server**:



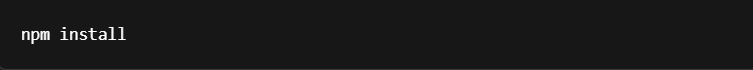
The frontend will be accessible at [**http://localhost:3000**](http://localhost:3000). The server supports **hot-reloading**, so changes are reflected instantly.

**2. Backend Setup**

1. **Navigate to the server directory**:



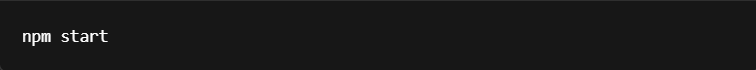
1. **Install Dependencies**:



1. **Set up Environment Variables**:

Create a .env file and add necessary configuration (e.g., MongoDB URI, JWT secret).

1. **Start the Backend Server**:



The backend will run at [**http://localhost:5000**](http://localhost:5000).

**3. Testing the Application**

* **Frontend**: Visit [**http://localhost:3000**](http://localhost:3000) to see the UI. Changes will reload automatically.
* **Backend**: Use tools like **Postman** to test API endpoints. Errors will appear in the terminal or browser console.

**4. Using a Proxy for Frontend-Backend Communication**

To avoid **CORS** issues, set up a proxy in the frontend's package.json:



This routes all API requests from the frontend to the backend.

**5. Stopping the Application**

* **Frontend**: Press Ctrl + C in the terminal.
* **Backend**: Press Ctrl + C in the backend terminal.

**6. Optional: Running in Production Mode**

For production, build the frontend and serve it alongside the backend. You can deploy to platforms like **Heroku** or **AWS**.

**7. API Documentation**

HouseHunt’s backend exposes a RESTful API built with **Node.js** and **Express.js**, allowing seamless communication with the frontend. All API endpoints return JSON responses and are secured using JWT where required.

**User Routes**

**➤ POST /api/users/signup – Register a New User**

Registers a new user and returns a JWT for authenticated access.

**Request Body:**



**Response:**



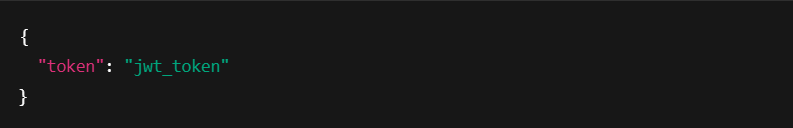
**➤ POST /api/users/login – Authenticate Existing User**

Validates credentials and returns a JWT if successful.

**Request Body:**



**Response:**

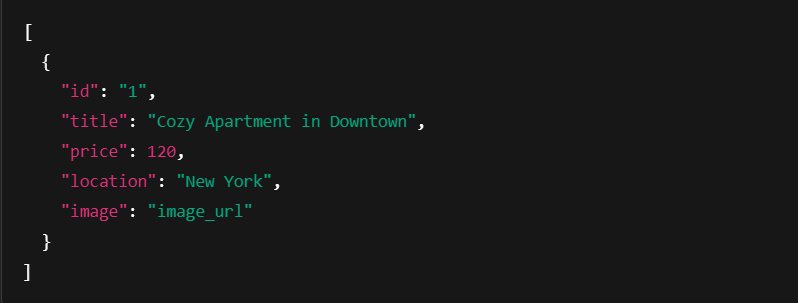


**Property Routes**

**➤ GET /api/properties – Get All Property Listings**

Returns a list of all properties available in the database.

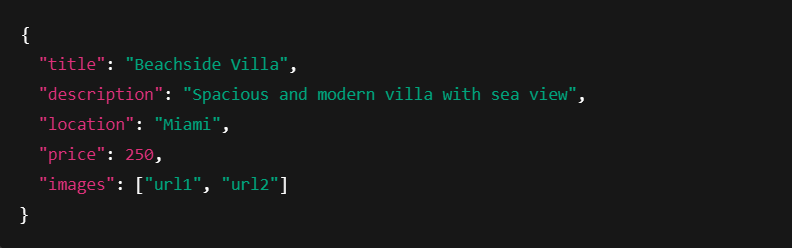
**Response:**



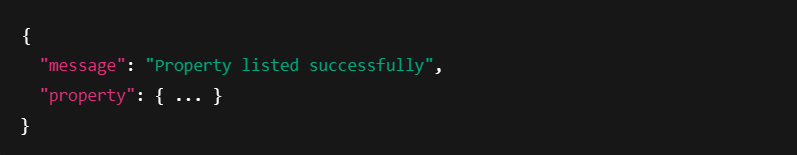
**➤ POST /api/properties – Add New Property (Protected)**

Allows a logged-in user (typically a host) to create a new property listing. Requires JWT in the header.

**Request Body:**



**Response:**



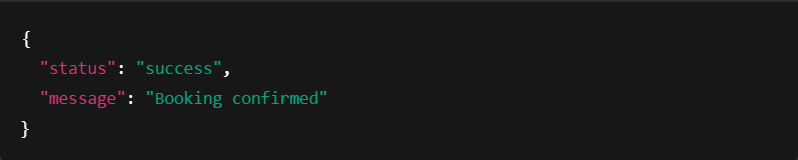
**Booking Routes**

**➤ POST /api/bookings – Book a Property (Protected)**

Allows an authenticated user to book a listed property.

**Request Body:**

**Response:**



**➤ GET /api/bookings/:userId – Get User Bookings (Protected)**

Retrieves all bookings made by a specific user.

**8. Authentication**

**HouseHunt** uses **JWT (JSON Web Tokens)** for secure authentication and authorization. Upon successful signup or login, users receive a JWT, which is stored in the browser's **localStorage**.

* This token is attached to subsequent API requests via the **Authorization header**.
* The backend verifies the token for protected routes such as creating properties or making bookings.
* Unauthorized or expired tokens are automatically rejected, and users are redirected to log in again.

Authentication middleware in Express ensures that sensitive endpoints are secure and accessible only to valid, logged-in users. This setup helps prevent unauthorized access, data breaches, and malicious activity.

**9. User Interface**

The **UI/UX** of HouseHunt is carefully designed with the end user in mind. Built with **Material-UI (MUI)** and **React**, the interface is responsive, fast, and easy to navigate.

**Key UI Features:**

* **Responsive Design**: Fully mobile-compatible layout using MUI's grid system and breakpoints.
* **Search Functionality**: Users can filter properties by **location**, **price range**, and **availability**.
* **Property Details View**: Detailed property pages display **images**, **descriptions**, **amenities**, **booking calendar**, and pricing.
* **User Dashboard**:
  + View and edit profile information
  + Track current and past bookings
  + Manage listed properties (for hosts)
* **Booking System**:
  + Users can pick check-in and check-out dates.
  + Real-time availability check.
  + Clear confirmation after booking.

The UI follows a **minimal and modern design**, ensuring both aesthetics and usability.

**10. Testing**

We prioritize code quality and reliability by implementing thorough testing on both the frontend and backend.

**🔹 Frontend Testing**

* **Tools Used**: Jest, React Testing Library
* **Test Coverage**:
  + Login and registration form behavior
  + Property card rendering
  + Booking flow interaction
* **User Interaction Testing**: Simulate real-world actions such as clicking buttons, entering data, and submitting forms to verify UI responses.

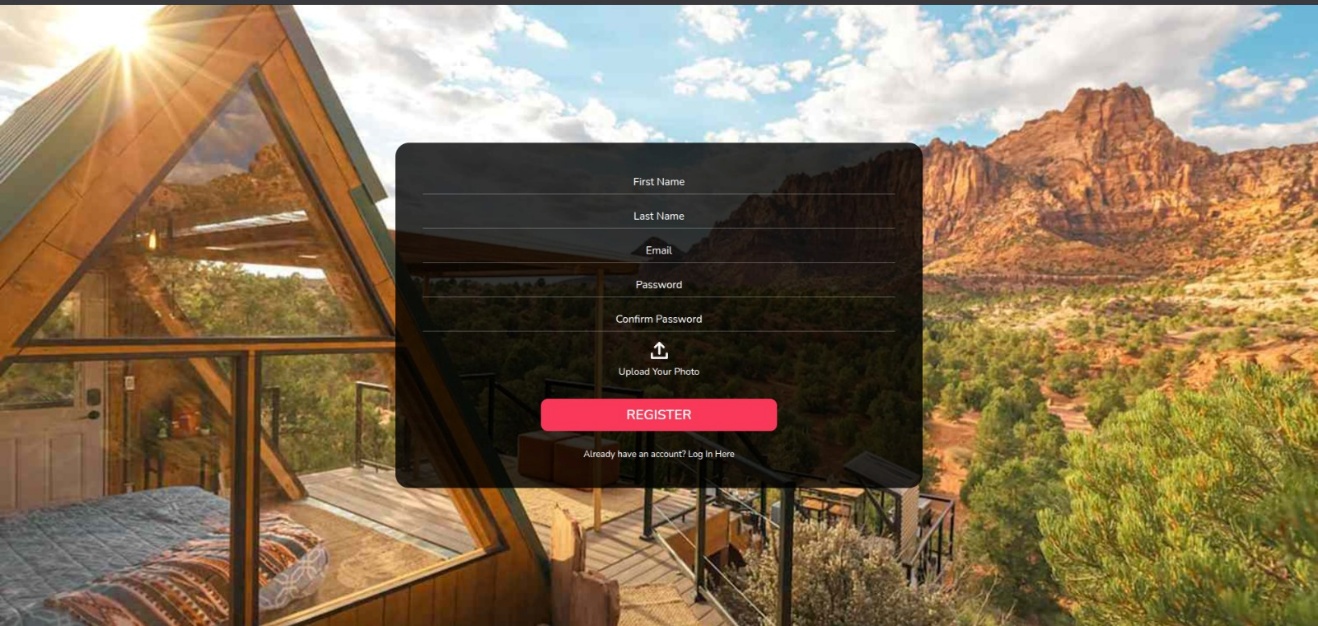
**🔹 Backend Testing**

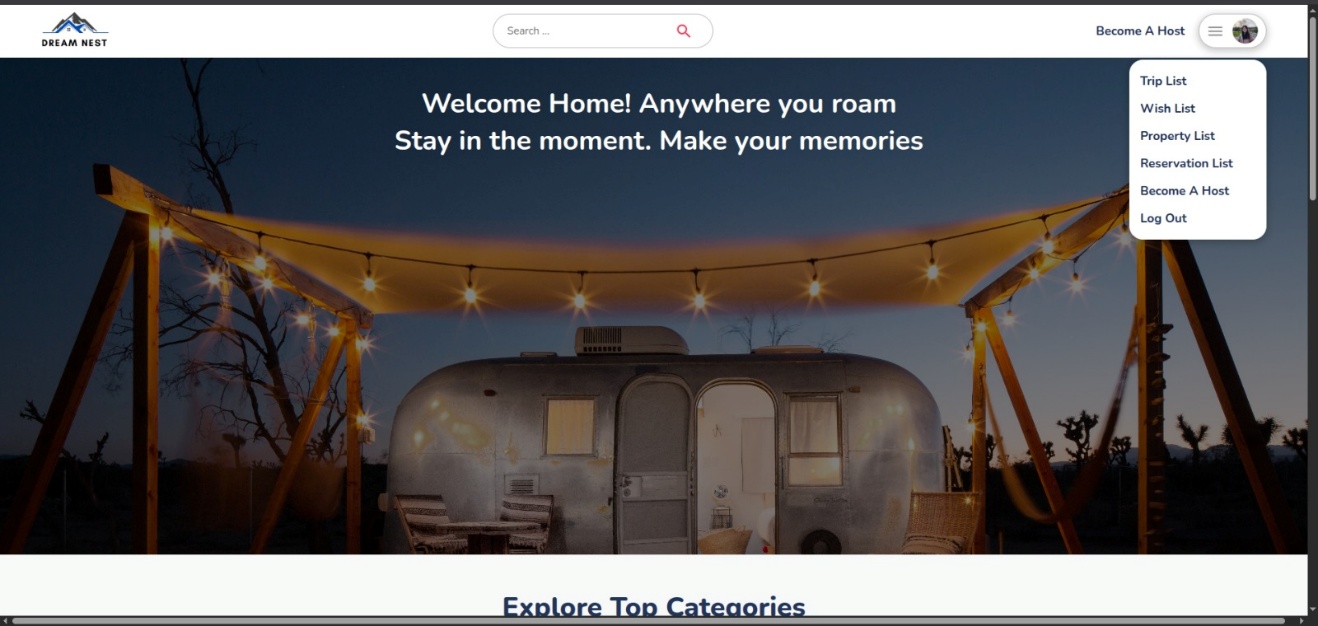
* **Tools Used**: Mocha, Chai, Supertest
* **API Endpoint Tests**:
  + User authentication (login/signup)
  + CRUD operations for properties
  + Booking requests with both valid and invalid data
* **Validation**: Tests also ensure proper validation messages are returned when required fields are missing or invalid.

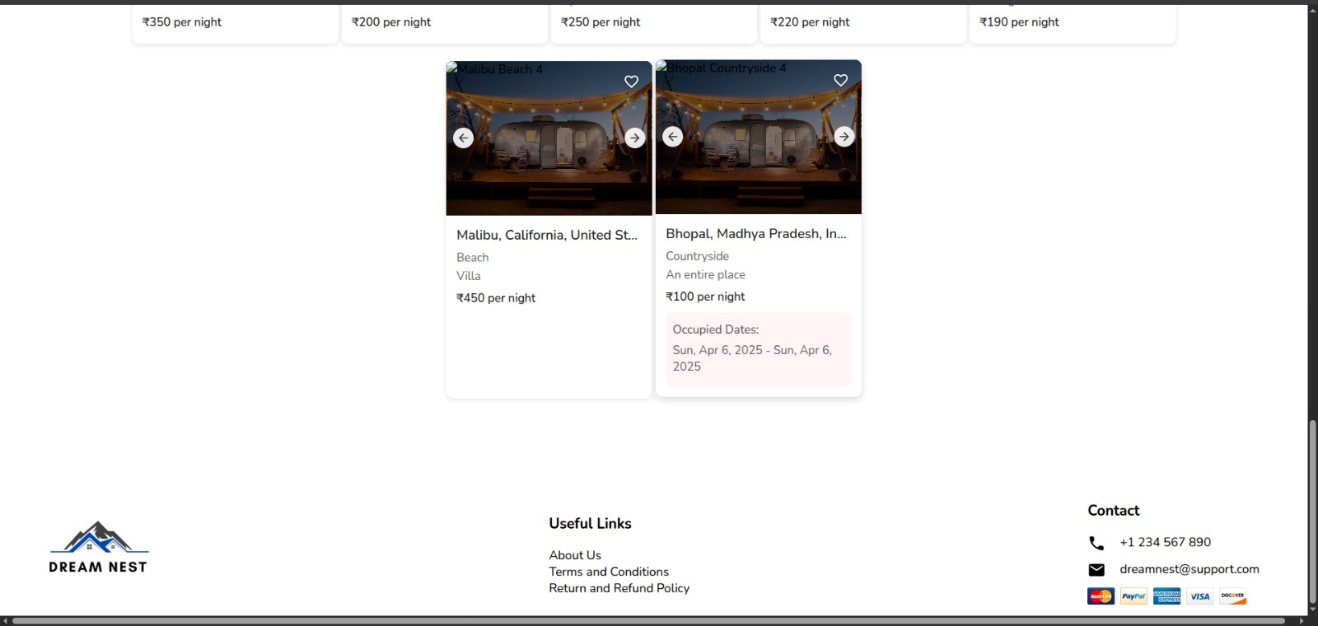
**11. Screenshots or Demo**

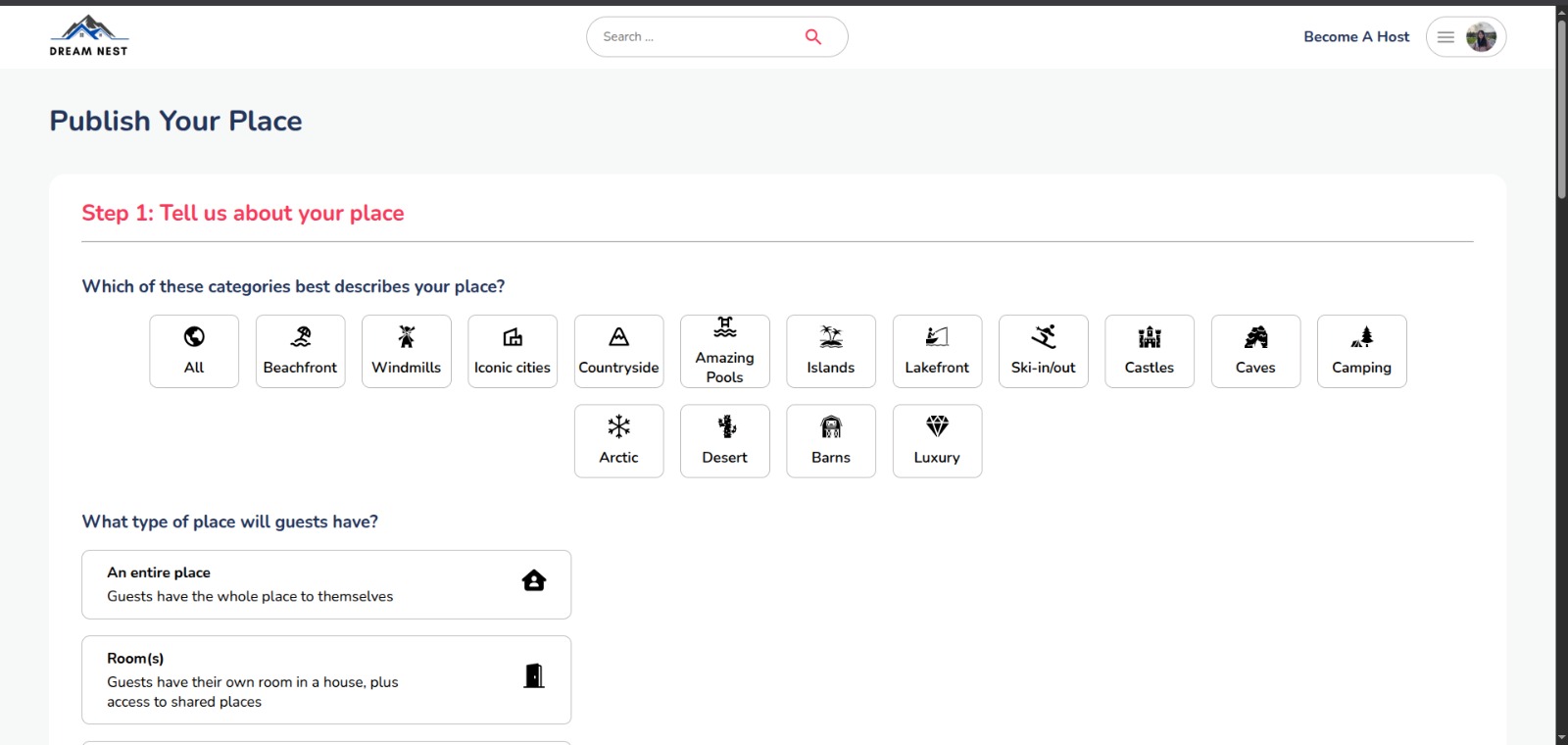
To better illustrate the functionality and visual design of HouseHunt, below are key interface screenshots and demo highlights that showcase the platform in action.

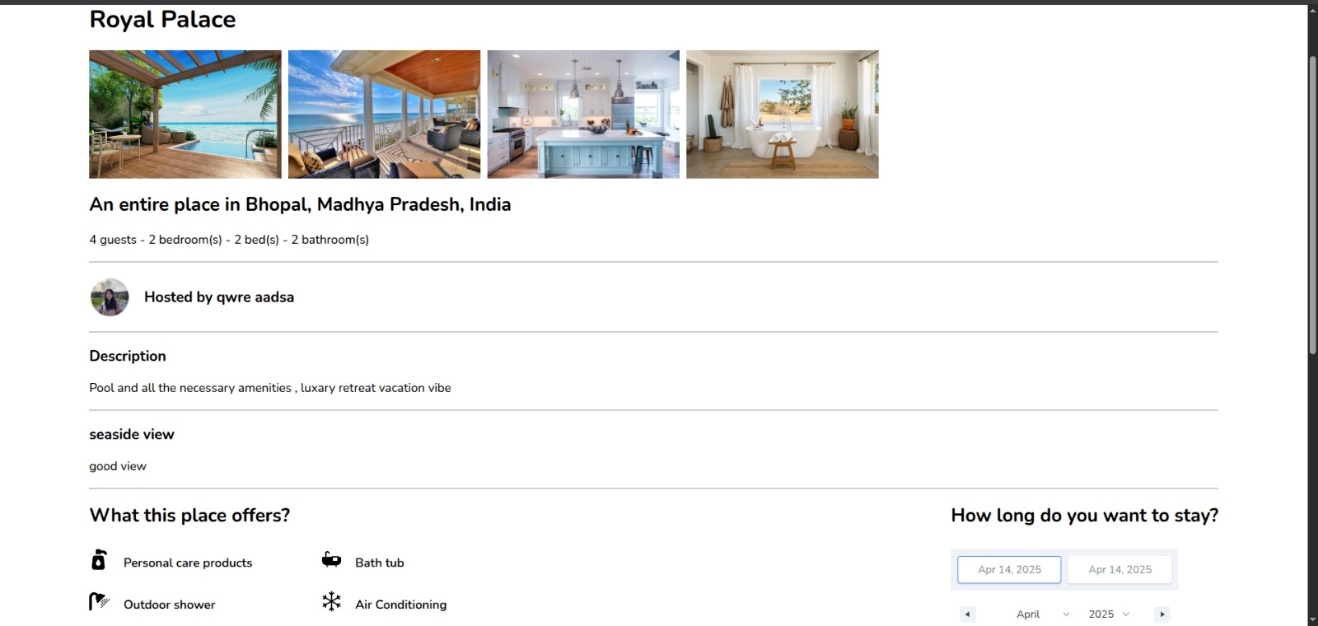
* **Home Page**
* Clean landing page with property highlights, search bar, and call-to-action for sign-up or listing properties.
* **Property Listings Page**
* Grid view of available properties with images, title, price, location, and quick view buttons.
* **Property Details Page**
* Individual property description, photo gallery, location map, pricing, and booking calendar.
* **User Dashboard**
* Profile management, booking history, and host listing controls (if user is a host).
* **Booking Confirmation**
* Final step showing confirmed dates and payment summary after a successful booking.

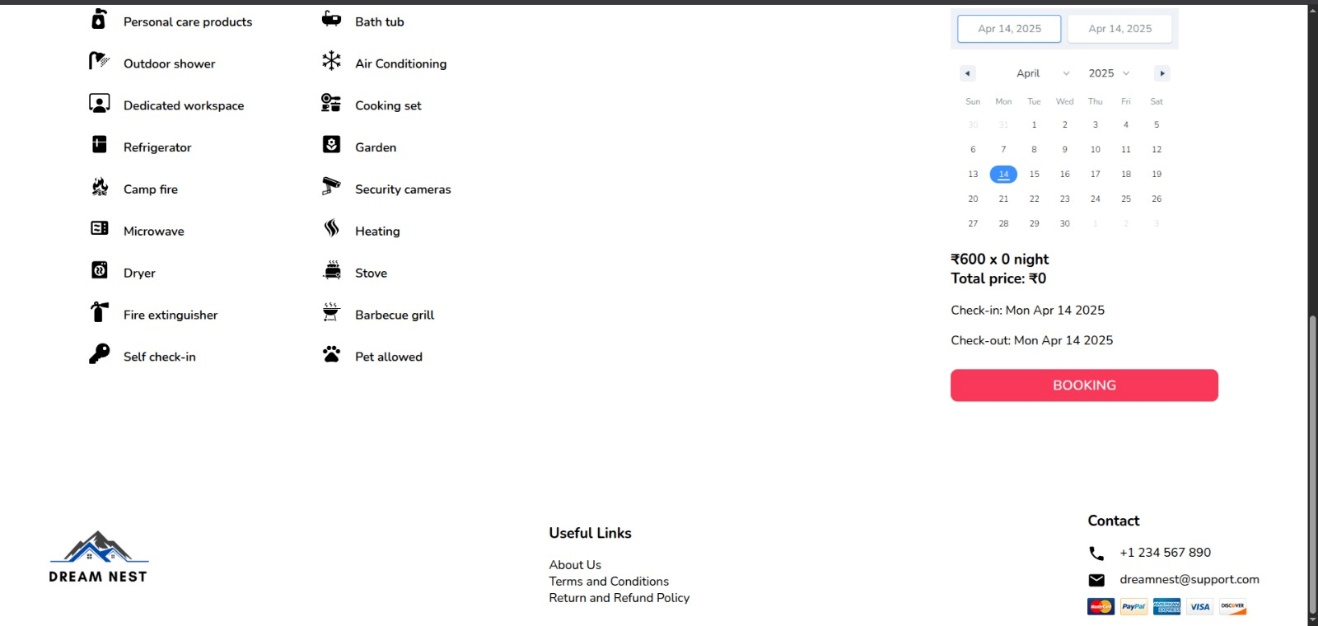


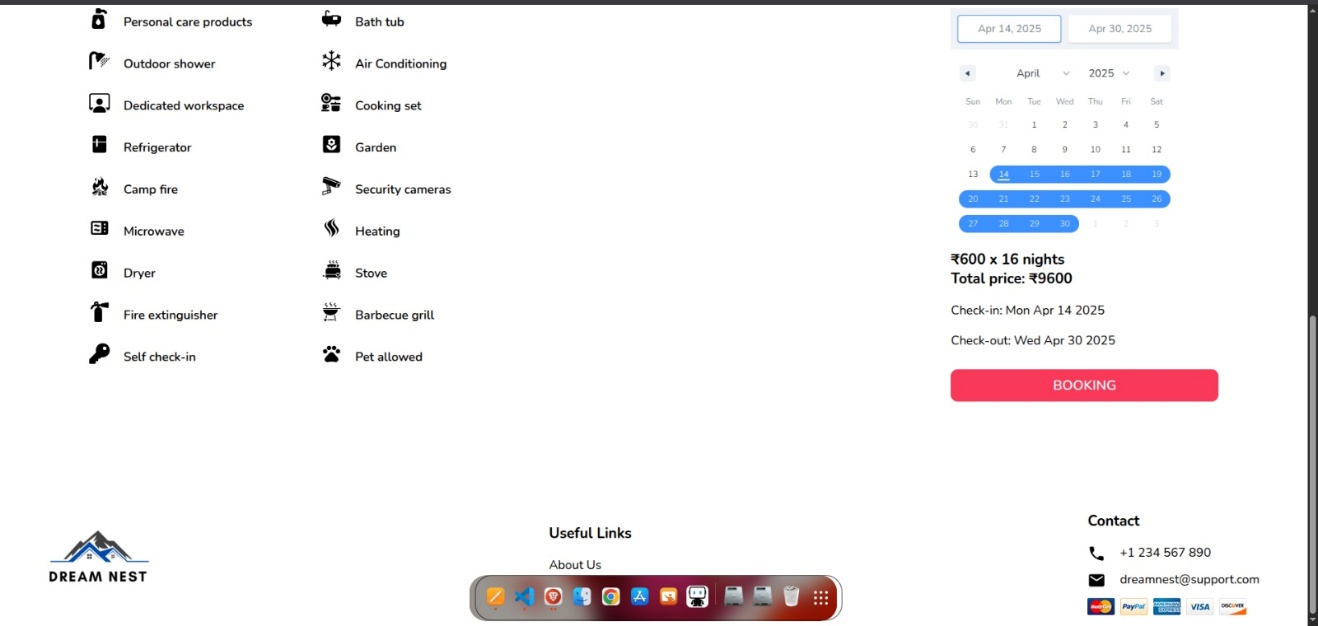


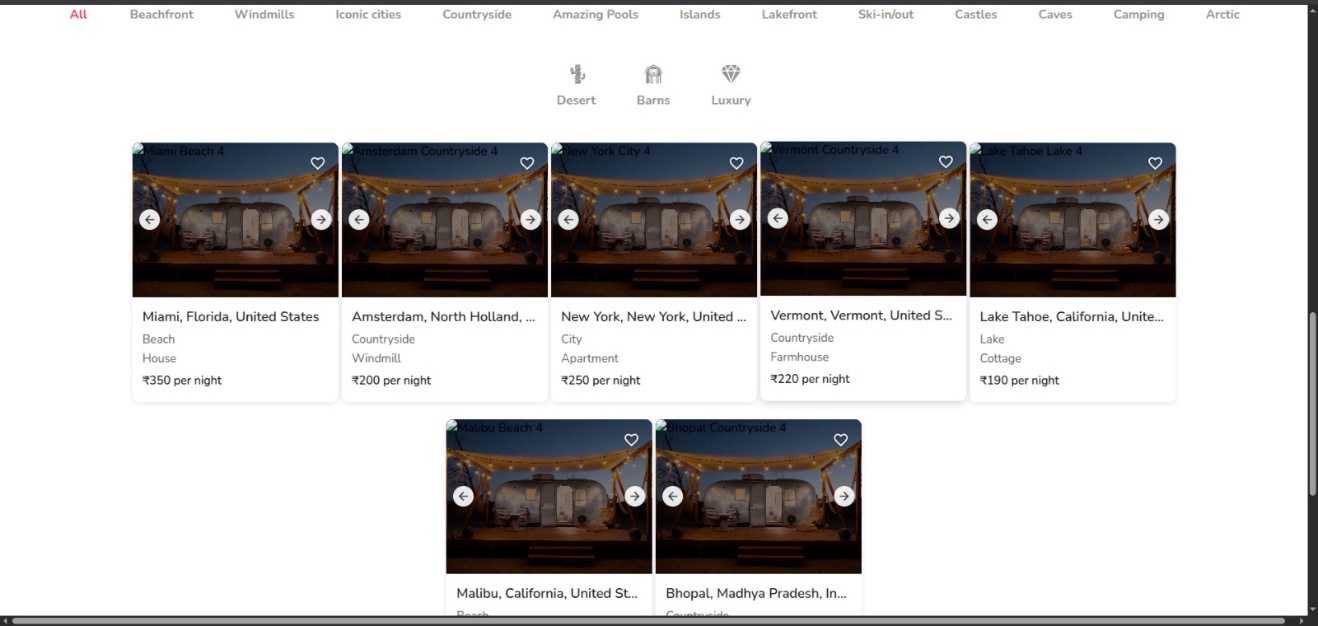


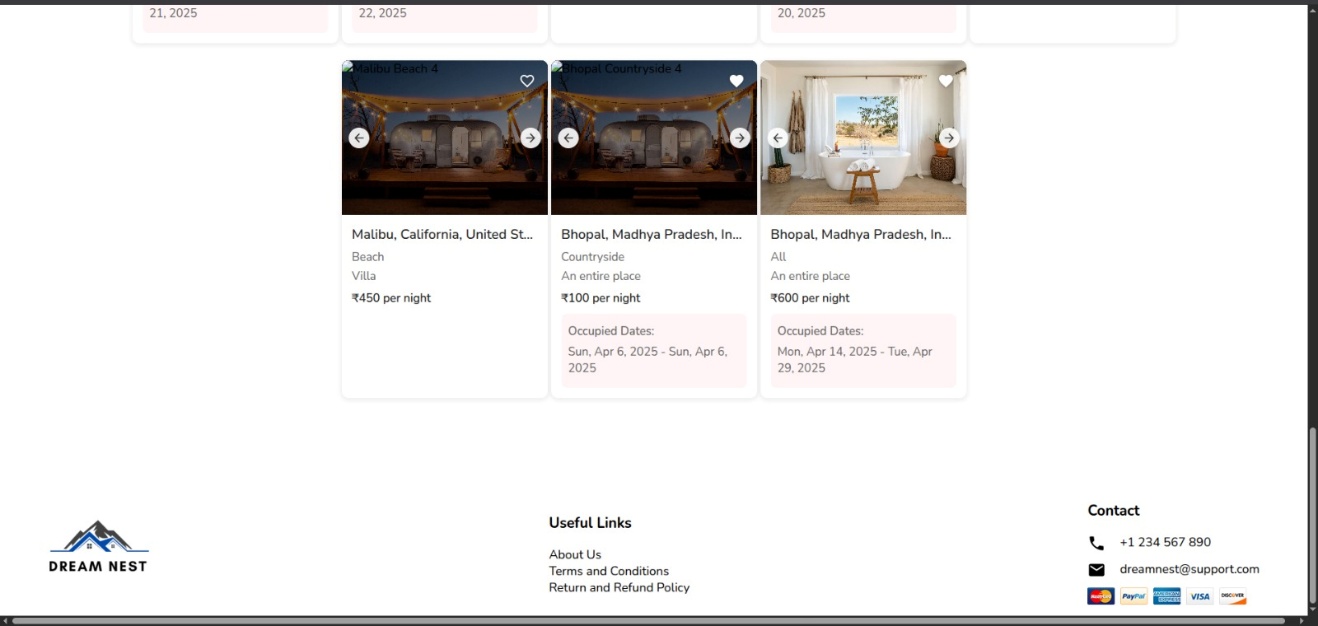


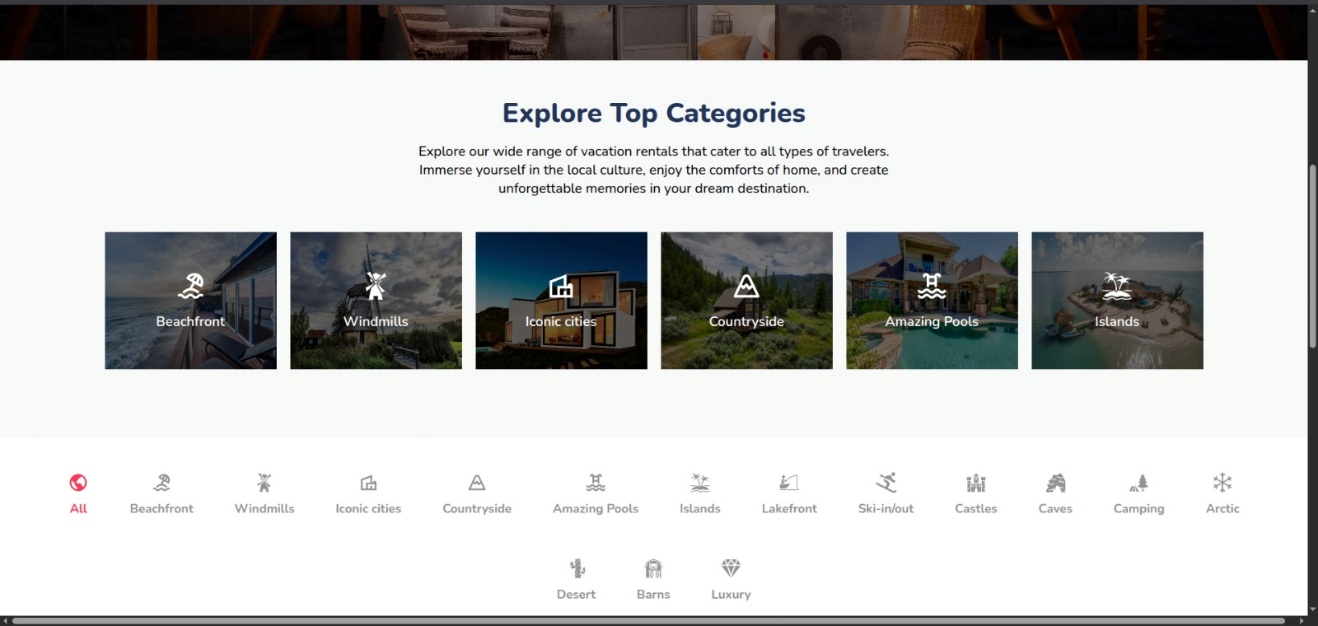












**12. Known Issues**

While HouseHunt is a fully functional property booking platform, there are still a few known issues that we're actively working to improve. These limitations do not significantly affect the core usability but are important to address in future iterations.

**1. Authentication Gaps**

* Currently, there's no token refresh mechanism; users must re-login once their session expires.
* Password reset and recovery features have not yet been implemented. If a user forgets their credentials, manual intervention is needed.

**2. Image Upload Limitations**

* Image upload lacks validation for maximum size and file type.
* Multiple image uploads for a single property listing are not yet supported.
* Upload failure messages are not always descriptive on poor networks.

**3. Booking Conflicts**

* Overlapping bookings are possible due to the absence of strict server-side date validation.

**4. UI/UX Challenges**

* Certain components like modals and calendars may not render optimally on smaller screens.
* The mobile navigation bar may overlap with content on low-resolution devices.

**5. General Error Handling**

* Some backend errors return generic messages, which can affect debugging and user feedback.
* Frontend does not yet handle all backend failure scenarios gracefully.

**13. Future Enhancements**

Here are some enhancements and features we’re planning to implement in future versions of HouseHunt to enrich the user experience and bring the platform closer to production quality:

To improve the functionality, scalability, and overall user experience of HouseHunt, we have identified several key areas for future development. These improvements are planned for future versions of the application.

**Feature Enhancements:**

**1. Secure Online Payments**

* Integration with payment gateways like Stripe or Razorpay to support direct online transactions.
* Functionality for issuing invoices, refunds, and tracking transaction history.

**2. Messaging System**

* Implementation of an internal chat system between hosts and guests using real-time technologies like Socket.io or Firebase.

**3. Ratings and Reviews**

* After completing a stay, users can provide feedback through ratings and written reviews.
* Ratings will help improve transparency and trust among the user base.

**4. Map Integration**

* Display property locations using Google Maps or Mapbox.
* Enable users to search for listings based on location proximity or nearby landmarks.

**5. Advanced Filtering Options**

* Filters by property type, price range, amenities (e.g., WiFi, AC), and guest capacity.
* Sorting options for popularity, ratings, or distance.

**6. Email Notifications**

* Sending booking confirmation emails with check-in details and cancellation policies.
* Weekly property suggestions based on user activity.

**7. Admin Dashboard**

* A dedicated interface for administrators to manage users, listings, and bookings.
* Dashboard features may include analytics, flagging inappropriate content, and platform monitoring.

**Technical Enhancements**

**1. JWT Token Refresh**

* Implementation of refresh tokens to maintain sessions without constant re-authentication.

**2. Role-Based Access Control (RBAC)**

* Differentiating features and permissions for Admins, Hosts, and Guests.
* Restricting access to certain routes and features based on roles.

**3. Performance Optimization**

* Adding pagination to listing pages to reduce load time.
* Lazy loading images and implementing code splitting for faster rendering.

**4. CI/CD Pipeline**

* Setting up Continuous Integration and Deployment using GitHub Actions or similar tools.
* Streamlining updates through automated testing and deployment.

**Conclusion**

HouseHunt stands as a practical and well-rounded full-stack web application that mirrors the functionality of leading property rental platforms while providing a simplified and tailored user experience. From a robust user authentication system to dynamic property listings and a seamless booking process, the project showcases the capabilities of the MERN stack in solving real-world problems. The development of HouseHunt has not only reinforced our technical understanding of full-stack architecture but also given us valuable insights into project planning, feature integration, and collaborative development. While there are still areas for growth and optimization, the foundation laid by this project provides a strong base for future enhancements, scalability, and potential real-world deployment.