UrbanNest

House Rent Web Application

UrbanNest

Team Members

Surya V

Bhuvan R

Bhairav J Shah

Prateek Kumar H

Objective :

- The primary objective of our House Rent Web Application is to create a digital platform that simplifies and streamlines the process of finding and renting properties.

- By providing a user-friendly interface and comprehensive property listings, Our web application aims to enhance the experience for both landlords and tenants.

Abstract :

- This project aims to develop a web application that streamlines the process of finding and renting houses.

- The application will provide a user-friendly interface for tenants to search for available properties based on their preferences, such as location, budget, and amenities.

- Landlords can easily list their properties, manage bookings, and communicate with potential tenants through the platform.

Existing Systems

Challenges:

- Mobile Responsiveness.
- Limites Search Optimization.
- Security Concerns.

Existing Systems

Possible Solutions:

- Mobile Application Development - By developing native apps for iOs and Android to improve accessibility and user experience.
- Enhanced Search and Recommendations - By implementing AI/ML algorithms to suggest properties based on user preferences and browsing history.
- Improved Security Measures - By using HTTPS for all connections, enforce secure password policies, and implement two-factor authentication(2FA).

Proposed System

Overview:

The proposed House Rent Web Application is designed to revolutionize the traditional process of finding and renting properties. This innovative platform aims to streamline the entire rental process, making it more efficient and convenient for both landlords and tenants.

Key Features:

1. **Comprehensive Property Listings:**
    - A vast database of verified property listings, categorized by location, budget, and property type.
    - Advanced search and filter options to help users find their ideal property.
    - Detailed property information, including photos, videos, and virtual tours.

2. **User-Friendly Interface:**

- An intuitive and visually appealing interface, accessible from any device.
- Easy navigation and seamless user experience.
- Personalized recommendations based on user preferences and browsing history.

3. **Landlord Dashboard:**

- A dedicated dashboard for landlords to manage their property listings, tenant inquiries, and rental agreements.
- Real-time analytics and insights to optimize rental strategies.

Front-end Architecture:Using React

The frontend of the Urban Nest Is built using React.js, a powerful library for creating dynamic and responsive user interfaces. It follows a component-based architecture to ensure reusability and maintainability.

Key Components and Features:

Component Hierarchy:

- **App Component:** Acts as the root component and manages the routing for the entire application.
- **Dashboard Component:** Displays a list of rental properties and provides filtering options.
- **Property Details Component:** Shows detailed information about a selected property.
- **User Profile Component:** Manages user-specific information such as bookings.
- **Admin Panel Component:** Manages user-specific information such as bookings.
- **Owner Panel Component:** Allows property owners to manage their listings.

Routing: React Routers is used to manage routes for different user roles (Renter, Owner, Admin) and app features, ensuring smooth navigation.

UI Libraries:

- **Material UI and Ant Design (Antd):** Enhance the UI with pre-built, responsive components.
- **Bootstrap:** Provides additional styling flexibility.

State Management:

- **Context API or Redux (Optional):** Used to manage global states such as user authentication and property data.

API Communication:

- **Axios:** Handles API requests to interact with the backend, fetching or submitting data efficiently.

UrbanNest Backend

Introduction

- **Objective**: Develop a scalable backend for a house rental platform.
- **Core Features**:
  - User Authentication
  - Property Listings
  - Search and Filter Functionality
  - Booking and Payments
  - Reviews and Ratings
- **Tech Stack**: MongoDB, Express.js, Node.js.

Backend Architecture Overview

- **Architecture**: RESTful API
- **Key Components**:
  - Authentication & Authorization
  - Data Management (MongoDB)
  - API Gateway (Express.js)
  - Payment Integration
  - Cloud Deployment (e.g., AWS/Heroku)
- **Third-party Integrations**: Stripe (Payments), Cloudinary (Image Uploads).

Database Design

Collections:

1. **Users**:
   - Schema: `{name, email, password, role (tenant/landlord), profilePicture}`
2. **Properties**:
   - Schema: `{title, description, location, price, landlordId, images, availability}`
3. **Bookings**:
   - Schema: `{propertyId, tenantId, startDate, endDate, status, paymentId}`
4. **Reviews**:
   - Schema: `{propertyId, userId, rating, comment}`

Relationships:

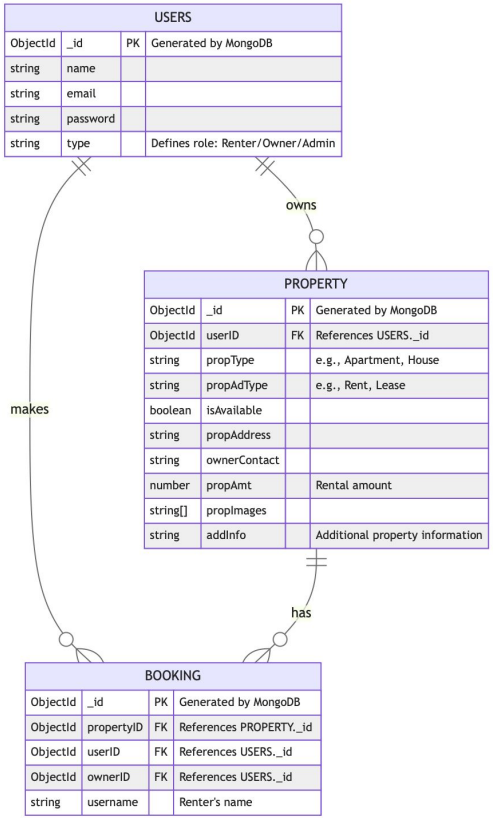
- Users ↔ Properties: One-to-Many
- Properties ↔ Bookings: One-to-Many
- Properties ↔ Reviews: One-to-Many

Authentication & Authorization

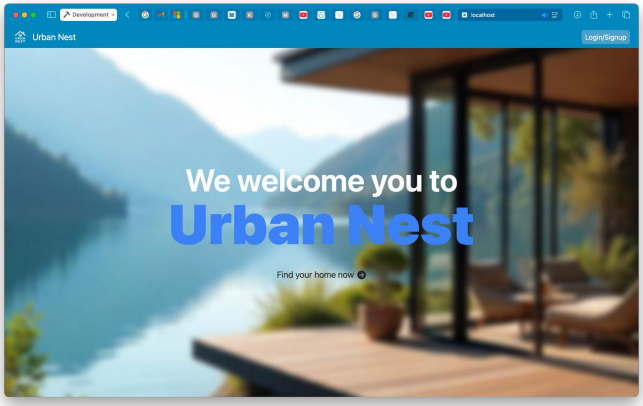
- **Authentication**:
  - Implemented using **JWT (JSON Web Tokens)**.
  - Secure storage of tokens in **HTTP-only cookies**.
- **Authorization**:
  - Role-based access control (Tenant vs. Landlord).
  - Example: Only landlords can create/edit properties.

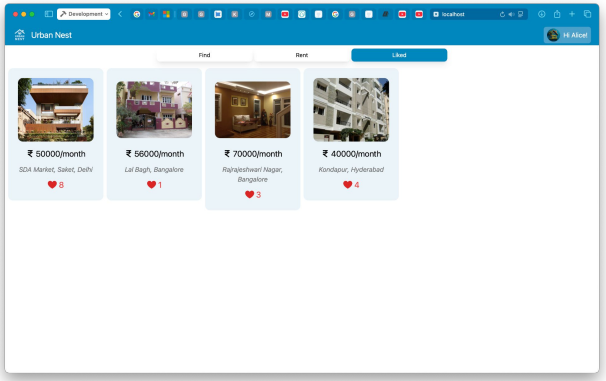
UrbanNest

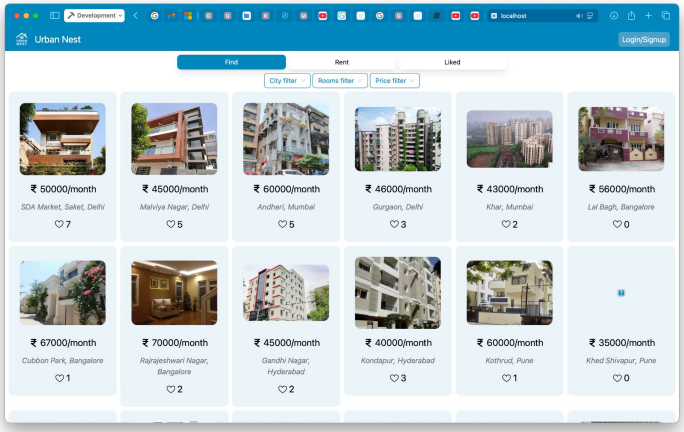
MongoDB Schema

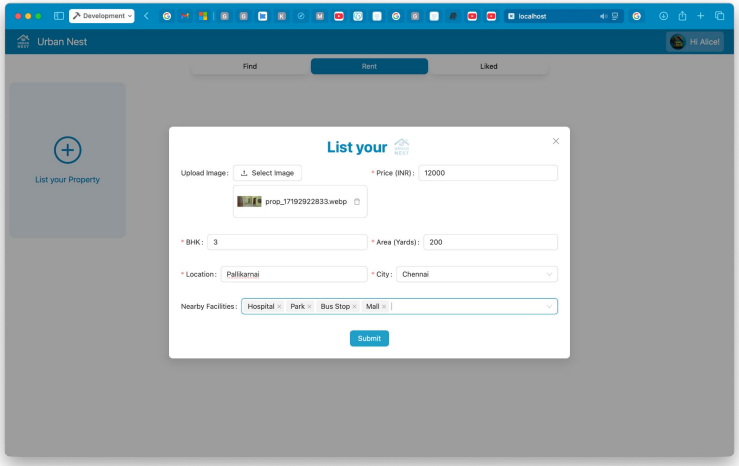


UrbanNest
Screenshots

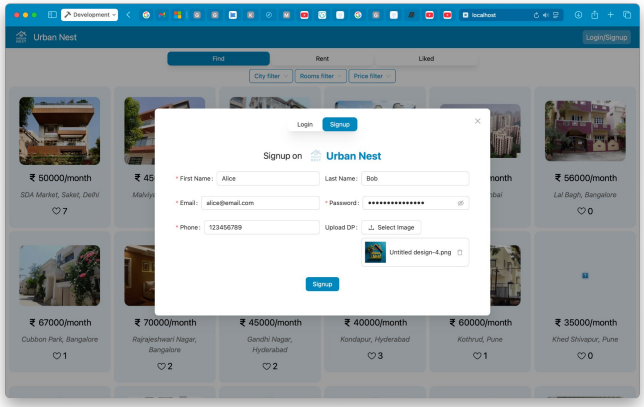


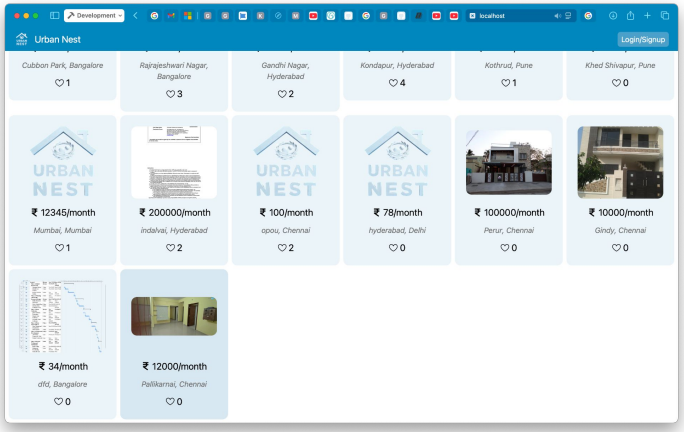


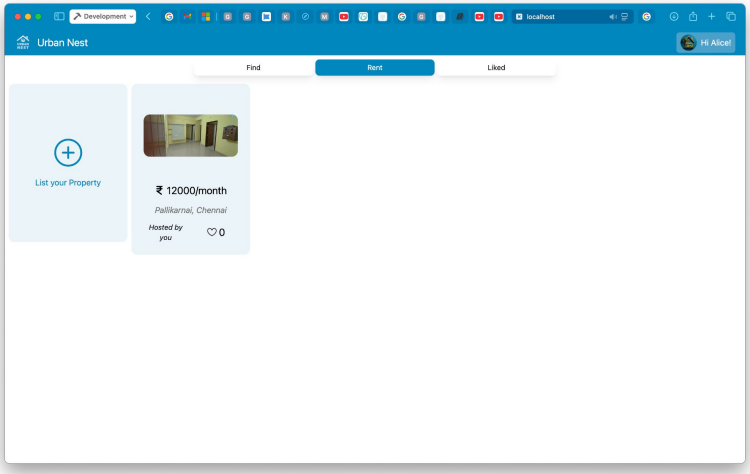


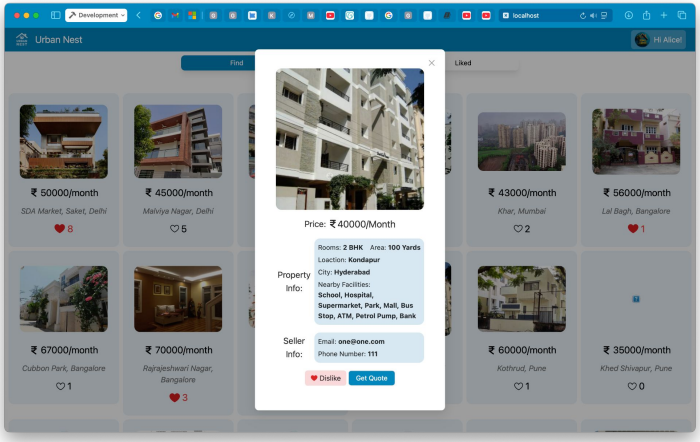


UrbanNest Screenshots









Requirements

Software:

- Node Package Manager/ Runtime
- Tailwind Server
- VScode or any editor with LSP support
- MongoDB Community Server

Hardware:

- An 64bit Arch, Linux or Unix Server with:
  - At least, dual core intel i5 processor or equivalent processor.
  - At least, 4 GB Unified/Un-unified primary memory(RAM).
  - At least, 10 GB RAID FS secondary memory storage.
  - At least, 2 Mbps Stable Internet connection with public IP access.
- An I/O Device to interact with server i.e. an laptop with ssh

Conclusion

In conclusion, **UrbanNest** exemplifies the potential of modern web technologies in creating efficient, user-friendly solutions for everyday challenges. As a comprehensive marketplace for renters and tenants in major Indian cities, it simplifies property transactions by bridging the gap between property owners and prospective tenants. Developed using the robust MERN stack and enhanced by tools like Tailwind CSS, Ant Design, Cloudinary, and EmailJS, UrbanNest provides a seamless experience for users to list, browse, and manage properties. Its deployment on Vercel ensures accessibility, while its open-source nature invites continuous learning and contributions. UrbanNest is a step toward making property rentals more streamlined and accessible.