HOUSE RENT APP USING MERN

INTRODUCTION:

PROJECT TITLE: House Rent App Using MERN

TEAM MEMBERS:

1. NEHA R – Frontend
2. LOGASRIMATHY S – Backend
3. JASMINE N - Frontend
4. NAVEEN RAJ K – Backend
5. GODWINRAJ T – Frontend

The House Rent App is a web-based platform built using the MERN stack (MongoDB, Express, React, Node.js) that connects property owners and tenants. It allows users to list available rental properties with details such as location, features, and pricing. Tenants can browse and filter properties based on their preferences for a seamless search experience. The app ensures efficient data management and provides dynamic user interfaces for property interactions.

PROJECT OVERVIEW:

PURPOSE:

The purpose of the House Rent App is to simplify the process of connecting property owners with potential tenants through an intuitive online platform. It provides a centralized space for property owners to list their rental properties, ensuring visibility to a wide audience. Owners can upload property details such as location, size, amenities, and pricing, making it easier for tenants to evaluate their options.

For tenants, the app offers a convenient way to search and filter properties based on criteria such as location, budget, and property type. This reduces the time and effort required to find suitable accommodations. The app bridges the gap between landlords and tenants by streamlining property listing details and contact information.

The use of the MERN stack ensures efficient data management and user-friendly interactions. React enables seamless user interfaces, while Node.js and Express handle backend processes effectively. MongoDB securely stores and manages property data, ensuring scalability as the user base grows.

The app's goal is to provide a hassle-free experience for both property owners and tenants, reducing reliance on traditional methods such as classified ads or physical visits. By offering detailed property information, the app promotes transparency and informed decision-making for renters.

FEATURES:

1. Property Listings

- Enables property owners to add rental properties with detailed descriptions and images.
- Listings include essential details such as size and available amenities.

2. Search and Filters

- Provides tenants with options to search for properties based on type.
- Advanced filters refine search results according to user preferences.

3. Property Details

- Each property has a detailed page displaying all relevant information, including images and specifications.
- Ensures tenants can explore properties thoroughly before making decisions.

4. User Authentication

- Offers secure login and signup for both property owners and tenants.
- Role-based access separates functionality for owners and tenants.

5. User Dashboard

- Owners can manage their listed properties in an organized dashboard.
- Tenants can mark properties as favorites for easy reference later.

6. Data Management

- Efficiently stores and organizes property and user information using a database.
- Ensures data accuracy and ease of access for property owners and tenants.

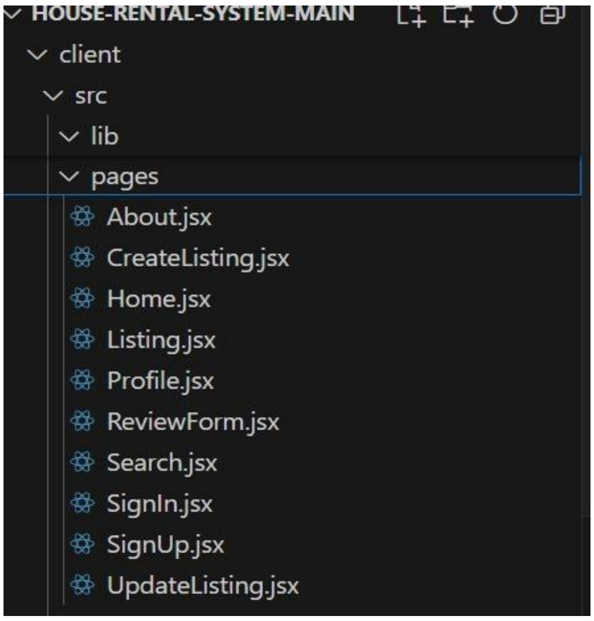
7. Secure Application

- Employs robust backend technologies to safeguard user data and property information.
- Follows best practices for maintaining user privacy and security.

ARCHITECTURE

Frontend (React Architecture)

The frontend is structured with React, utilizing a component-based architecture, which allows for reusability and modularity. The folders and files indicate a separation of concerns based on roles and common functionalities:



```
HOUSE-RENTAL-SYSTEM-MAIN
  client
    src
      lib
      pages
        About.jsx
        CreateListing.jsx
        Home.jsx
        Listing.jsx
        Profile.jsx
        ReviewForm.jsx
        Search.jsx
        SignIn.jsx
        SignUp.jsx
        UpdateListing.jsx
```

• **src/pages Folder**:

Contains React components representing individual pages of the application, organized by their functionalities and purpose.

- **About.jsx**:

  Provides general information about the application and its purpose, focusing on how it connects users with property listings.

- **CreateListing.jsx**:

  Allows property owners to create and publish new property listings with essential details like descriptions and images.

- **Home.jsx**:
  Serves as the landing page of the application, showcasing featured property listings and an overview of the platform.

- **Listing.jsx**:

  Displays basic details about a specific property listing, such as amenities and the owner's general contact information (excluding interactive elements).
- **Profile.jsx**:

A personal page for users to view and update their account information and preferences.

- **ReviewForm.jsx**:

  Enables users to leave reviews for properties they have used, contributing to the platform's review system without real-time interactions.

- **Search.jsx**:

  Provides a search interface for users to look up properties based on simple filters like property type, excluding location-based filtering or dynamic updates.

- **SignIn.jsx**:

  A page for existing users to log in to their accounts to access non-interactive features.

- **SignUp.jsx**:

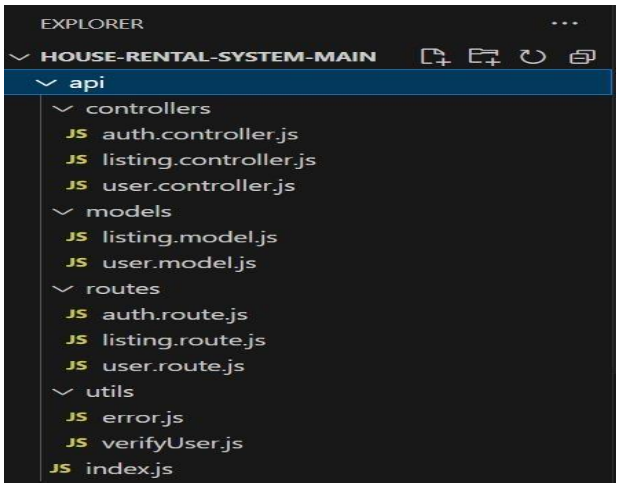
  Allows new users to register and create an account, focusing only on essential onboarding details.

- **UpdateListing.jsx**:

  Allows property owners to update their property listing details, such as descriptions or status, without dynamic pricing or advanced availability options.

Backend (Node.js and Express Architecture)

The backend architecture is built with Node.js and Express, providing RESTful
APIs for the frontend to interact with:



```
EXPLORER
HOUSE-RENTAL-SYSTEM-MAIN
  api
    controllers
      auth.controller.js
      listing.controller.js
      user.controller.js
    models
      listing.model.js
      user.model.js
    routes
      auth.route.js
      listing.route.js
      user.route.js
    utils
      error.js
      verifyUser.js
    index.js
```

index.js (Backend Entry Point)

1. **Express Server Setup:**

- Initializes the Express app.
- Defines essential routes for endpoints such as user authentication, order management, and menu retrieval.
- Middleware includes:
  - **JSON Parsing:** To handle incoming JSON payloads.
  - **CORS:** Enables secure cross-origin requests from the frontend.
  - **Error Handling:** Provides basic error responses for invalid routes or server errors.

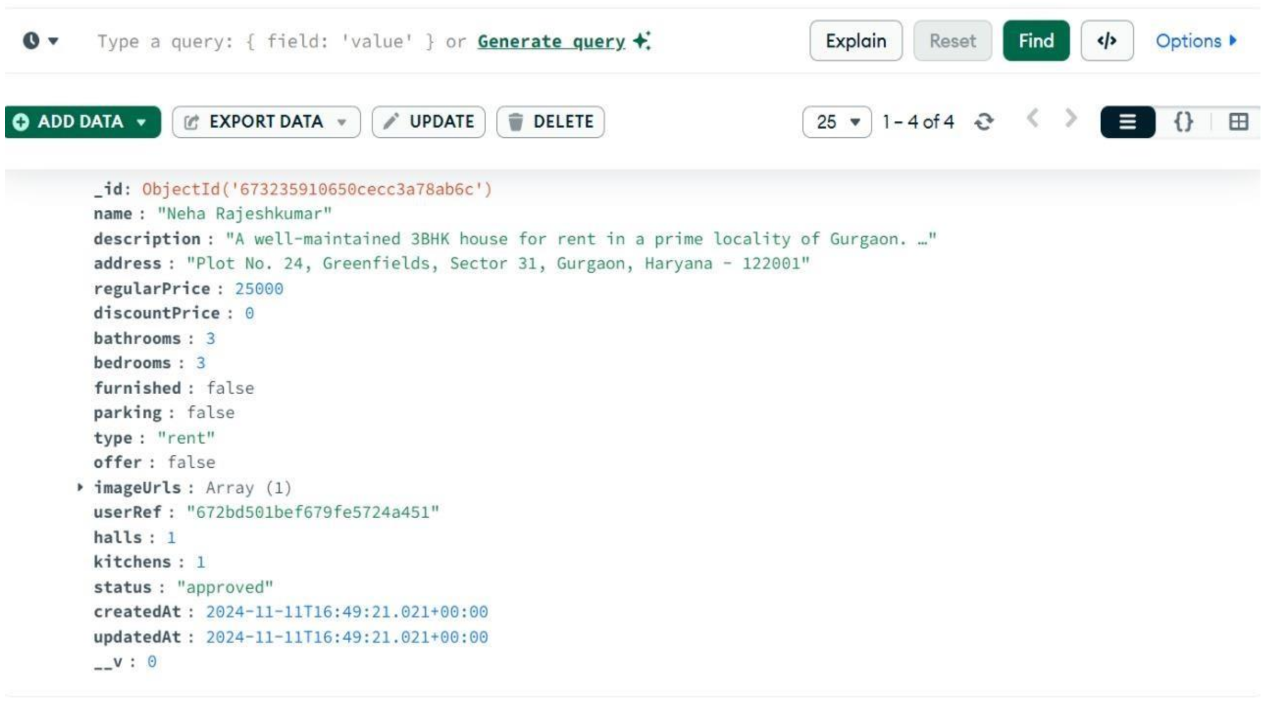
2. **Route Handling:**
   - **User Routes:** Handles authentication (login, registration) and user-specific operations.
   - **Menu Routes:** Serves menu items for the frontend.
   - **Order Routes:** Processes and retrieves orders.

config.js

This file centralizes configuration, allowing you to:

- Define **database URIs** for MongoDB.
- Store the **JWT secret key** for secure token generation.
- Enable environment-specific setups like development or production modes.

Database (MongoDB and Schema Design)



```
_id: ObjectId('673235910650cecc3a78ab6c')
name : "Neha Rajeshkumar"
description : "A well-maintained 3BHK house for rent in a prime locality of Gurgaon. …"
address : "Plot No. 24, Greenfields, Sector 31, Gurgaon, Haryana - 122001"
regularPrice : 25000
discountPrice : 0
bathrooms : 3
bedrooms : 3
furnished : false
parking : false
type : "rent"
offer : false
imageUrls : Array (1)
userRef : "672bd501bef679fe5724a451"
halls : 1
kitchens : 1
status : "approved"
createdAt : 2024-11-11T16:49:21.021+00:00
updatedAt : 2024-11-11T16:49:21.021+00:00
__v : 0
```

The database structure seems to be defined in **Schema.js**:

- **MongoDB with Mongoose**:
  - **Schemas** define the structure of collections in MongoDB, ensuring data consistency.

- The primary collections could include:
  - **Users**: Contains details like user credentials, roles (admin, agent, user).

Setup Instructions

1. Prerequisites

- **Node.js**: Required to run the server-side JavaScript - download from Node.js officialwebsite.
- **MongoDB**: Database for storing user and complaint information - download fromMongoDB official website .

2. Installation Steps

Step 1: Install Dependencies

Navigate to the backend and frontend folders separately to install their dependencies.

For **Backend**:

- cd backend
- npm install

For **Frontend:**

- cd frontend
- npm run dev

Step 3: Set Up Environment Variables

- In the backend, create a `.env` file inside the backend folder to store environment variables. Add the following variables, replacing the placeholders with your actual values:

Step 4: Start the Application

To start both the frontend and backend servers:

For Backend:

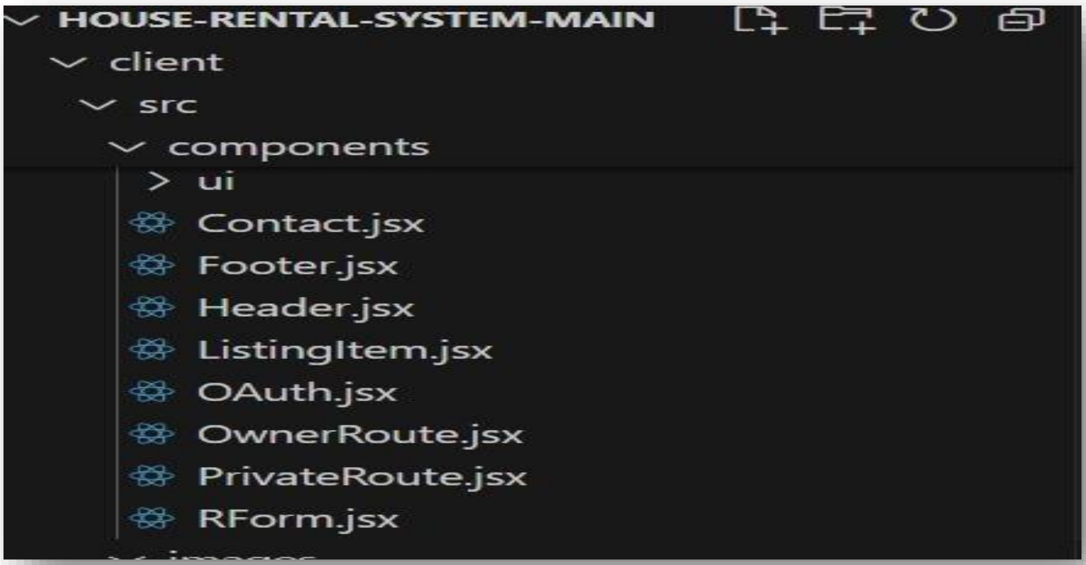
```
cd backend
npm start
```

For Frontend:

Client (Frontend)

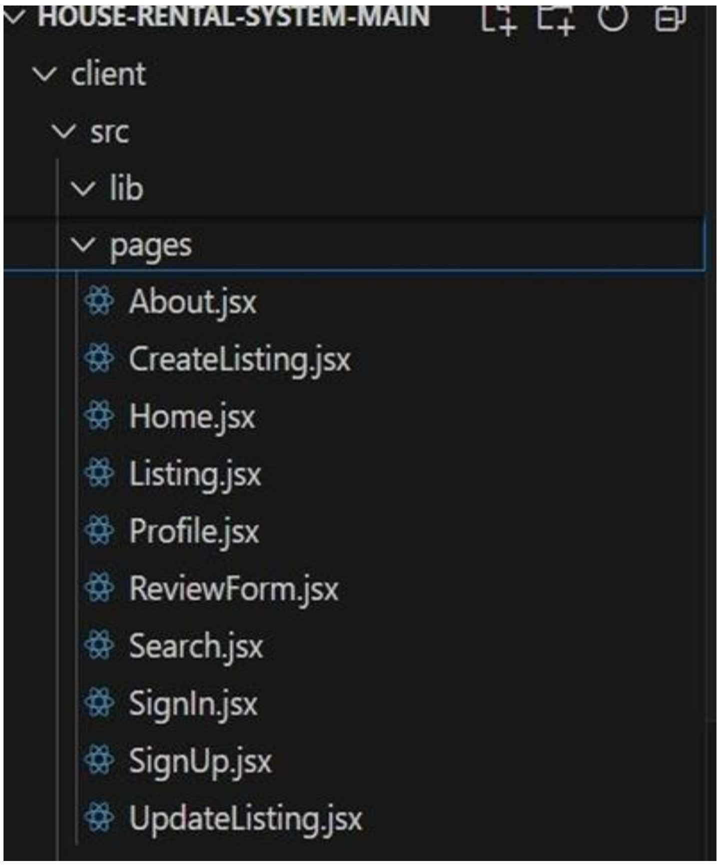
The React frontend structure is organized to separate different components, assets, and core files for easier maintainability. Here's a breakdown:

- **public/**: Contains the static assets, like index.html, which is the entry point of the React app. This file is used to load the app in the browser.
- **src/**: This is the main source folder where the React app's components, pages, and logic are organized.
  - **index.js**: The entry point of the React application. It renders the app and connects it to the root element in the index.html file.
  - **App.js**: The root component that defines the main structure of the application and manages routing.
  - **components/**: Contains all reusable React components, categorized based on their functionality.
    - **ui/**: Contains user interface components used across the application.
      - **Contact.jsx**: Component for displaying a contact form or contact details.
      - **Footer.jsx**: A shared footer component used throughout the app.
      - **Header.jsx**: The header or navigation bar component, present on all pages.



```
HOUSE-RENTAL-SYSTEM-MAIN
  client
    src
      components
        > ui
        Contact.jsx
        Footer.jsx
        Header.jsx
        ListingItem.jsx
        OAuth.jsx
        OwnerRoute.jsx
        PrivateRoute.jsx
        RForm.jsx
```

- **pages/:** Contains React components that represent individual pages of the app.
  - **About.jsx:** Provides information about the app or its purpose.
  - **CreateListing.jsx:** A page for property owners to create and publish new property listings.
  - **Home.jsx:** The landing page of the application, showcasing featured properties and search functionality.
  - **Listing.jsx:** Displays detailed information about a selected property listing.
  - **Profile.jsx:** A user profile page where users can manage their account details and preferences.
  - **ReviewForm.jsx:** A page for users to submit reviews about properties or landlords.
  - **Search.jsx:** A page with search functionality for finding properties based on user preferences.
  - **SignIn.jsx:** A page for users to log in to their accounts.
  - **SignUp.jsx:** A page for new users to register and create accounts.
  - **UpdateListing.jsx:** Allows property owners to update details of an existing property listing.

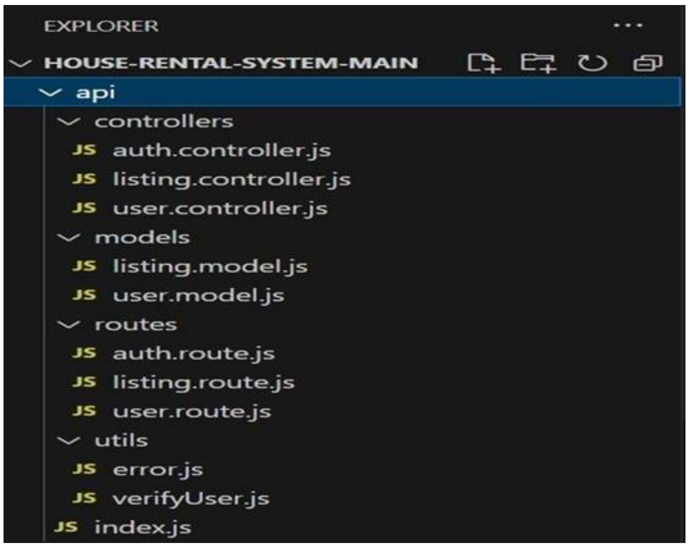


```
HOUSE-RENTAL-SYSTEM-MAIN
  client
    src
      lib
      pages
        About.jsx
        CreateListing.jsx
        Home.jsx
        Listing.jsx
        Profile.jsx
        ReviewForm.jsx
        Search.jsx
        SignIn.jsx
        SignUp.jsx
        UpdateListing.jsx
```

Server (Node.js Backend)
The Node.js backend is organized to separate server logic, configurations, and routes for a modular design. Here's a breakdown:

• **backend/**

- **index.js**: The main entry point of the backend server. It initializes the Express server, connects to MongoDB, and sets up middleware and routes.
- **package.json** and **package-lock.json**: Contain project metadata and dependencies for the backend server.
- **Schema.js**: Contains MongoDB schemas and models. This file defines the data structure for storing information in MongoDB, such as user and complaint details.
- **config.js**: Stores configuration settings like database connection strings, port numbers, and any secret keys (typically using environment variables). This file is imported in index.js and other backend files that require access to environment-specific values.
- **node_modules/**: Contains all the installed Node.js packages required by the backend.



```
EXPLORER
HOUSE-RENTAL-SYSTEM-MAIN
  api
    controllers
      auth.controller.js
      listing.controller.js
      user.controller.js
    models
      listing.model.js
      user.model.js
    routes
      auth.route.js
      listing.route.js
      user.route.js
    utils
      error.js
      verifyUser.js
    index.js
```

Running the Application

Frontend: Start the React frontend server.

- cd frontend
- npm run dev

Run this command in the frontend directory to launch the React app. It will start a development server, typically on http://localhost:3000.

Backend: Start the Node.js backend server.

- cd backend
- npm start

It usually runs on http://localhost:5000 or another port specified in your configuration.

API DOCUMENTATION

1. User Registration and Authentication

- **Endpoint**: /api/users/register
  - **Method**: POST
  - **Description**: Registers a new user.
  - **Request Body**:

```
{
  "name": "John Doe",
  "email": "johndoe@example.com",
  "password": "securePassword123"
}
```

AUTHENTICATION

In this project, authentication and authorization are implemented to securely manage user access to various resources and actions within the complaint registry system. Here's an overview of how these processes work:

1. **Authentication Method**

   JWT (JSON Web Token) Authentication
   Authentication Using JWT: • Authentication is handled with JSON Web Tokens (JWTs). Upon user login, a token is generated and sent to the client. This token is used to authenticate future requests without requiring repeated logins.

   JWT Workflow:

   1. The user logs in by providing valid credentials (e.g., email and password) to the /login endpoint.
   2. The backend generates a JWT containing user information (e.g., user ID and role) upon successful login and sends it to the client.
   3. The client stores the JWT securely, typically in local storage or a secure cookie.
   4. For authenticated requests, the client includes the JWT in the HTTP header (Authorization: Bearer <token>).

   Token Structure:

   • **Header**: Specifies the token type (JWT) and signing algorithm (e.g., HS256).
   • **Payload**: Contains claims, such as user details and custom data like role and expiration time.
   • **Signature**: Validates the token's authenticity and ensures it hasn't been tampered with.

2. **Token-Based Authorization**

   Role-Based Access Control (RBAC):

   • The system supports role-based permissions for users, agents, and admins.
   • **Role-Specific Permissions**:
   o User: Can register, log in, and access personal resources.
   o Agent: Can manage assigned tasks and update statuses.
   o Admin: Manages system-level privileges, such as viewing and assigning roles.

   Implementation Details:

   • Protected routes validate the JWT token from the request header.
   • The token is decoded to extract user details and role.
   • Middleware enforces role-based access by ensuring users only access permitted endpoints and actions.

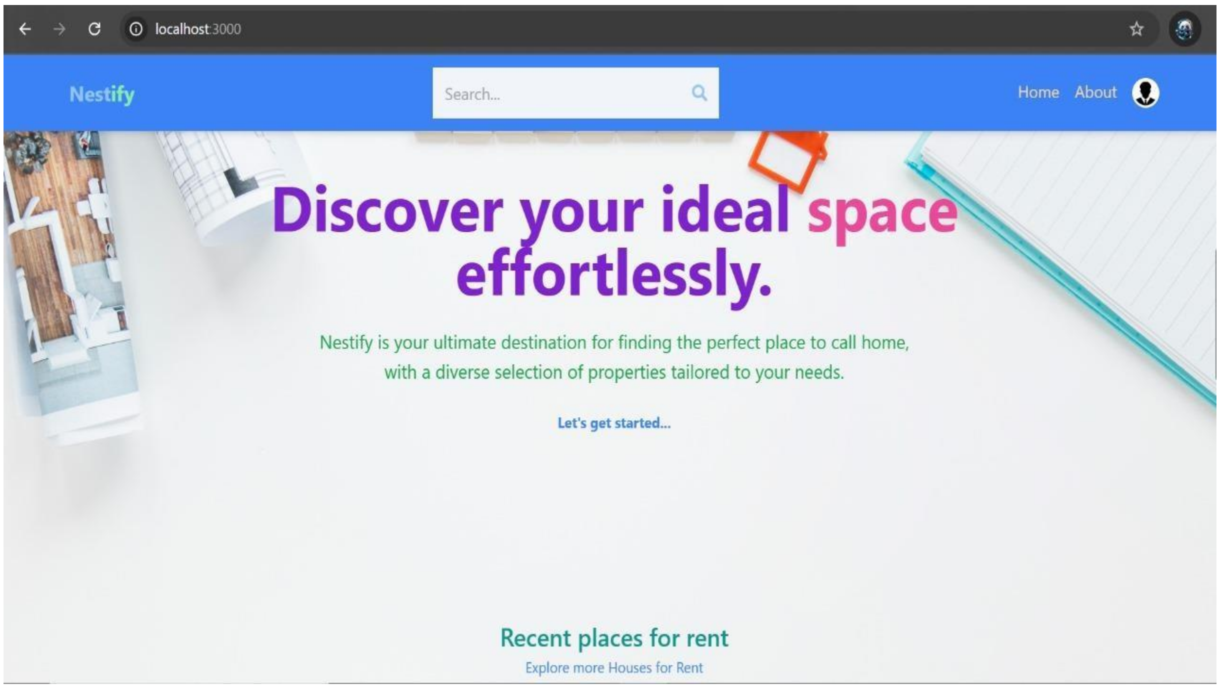
3. **Middleware for Authentication and Authorization**

   Authentication Middleware: • Verifies the presence and validity of the JWT in each request.
   • If the token is invalid or missing, access is denied, and an error response is returned.

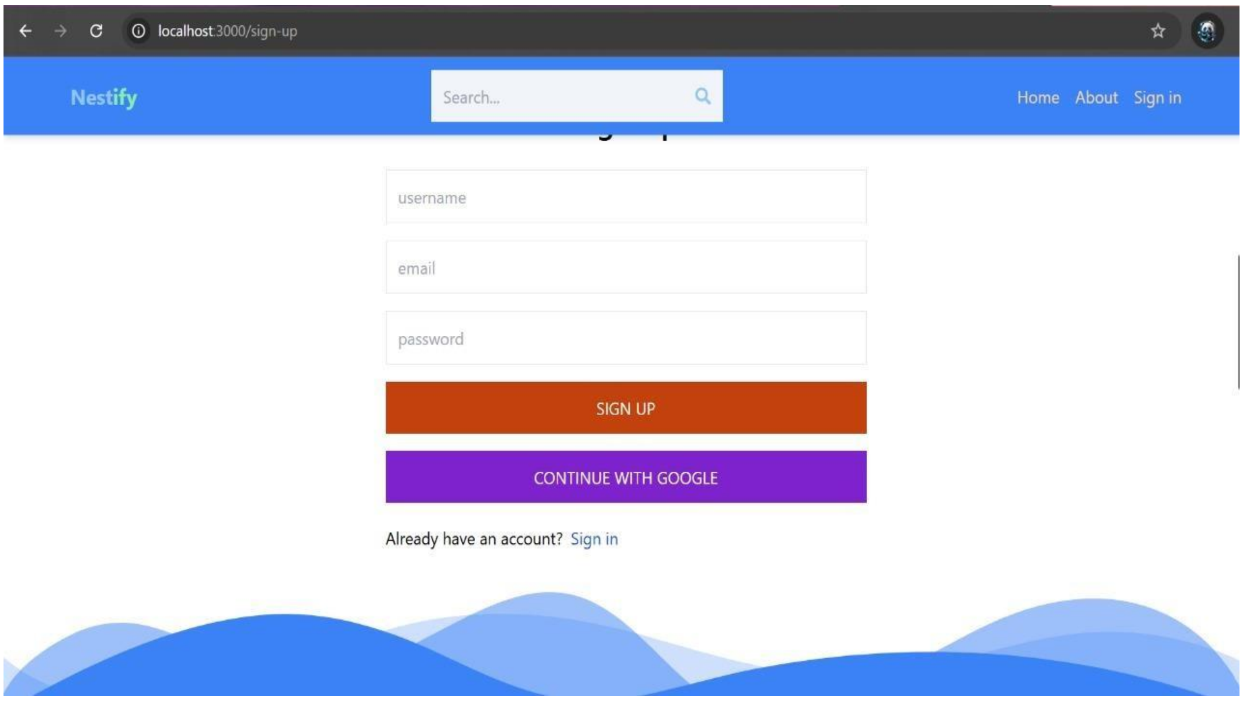
   Authorization Middleware: • Decodes the JWT to determine the user's role and permissions.
   • Allows or denies access based on the requested action and role, ensuring compliance with access policies.

USER INTERFACE

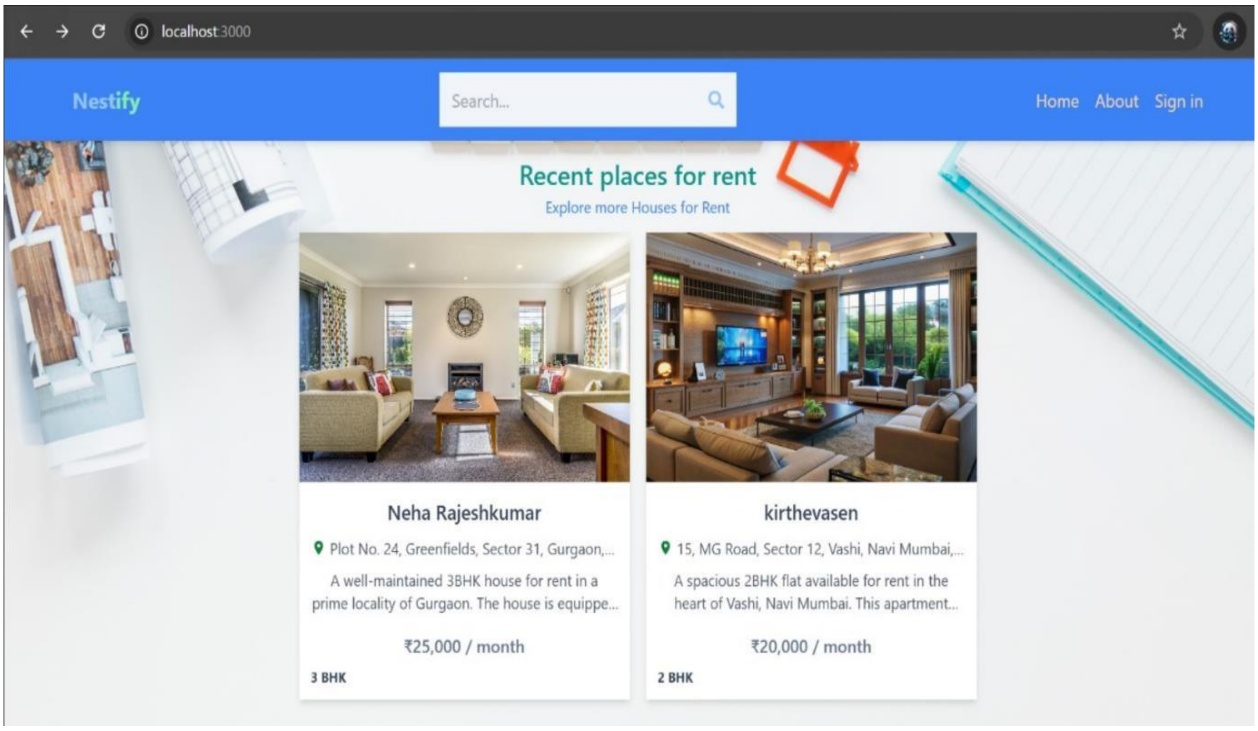
HomePage:


localhost:3000
Nestify
Search...
Home
About
Discover your ideal space effortlessly.
Nestify is your ultimate destination for finding the perfect place to call home, with a diverse selection of properties tailored to your needs.
Let's get started...
Recent places for rent
Explore more Houses for Rent

SignUp Page:



Property Detail Page:



Neha Rajeshkumar

Plot No. 24, Greenfields, Sector 31, Gurgaon,...

A well-maintained 3BHK house for rent in a prime locality of Gurgaon. The house is equippe...

₹25,000 / month

3 BHK

kirthevasen

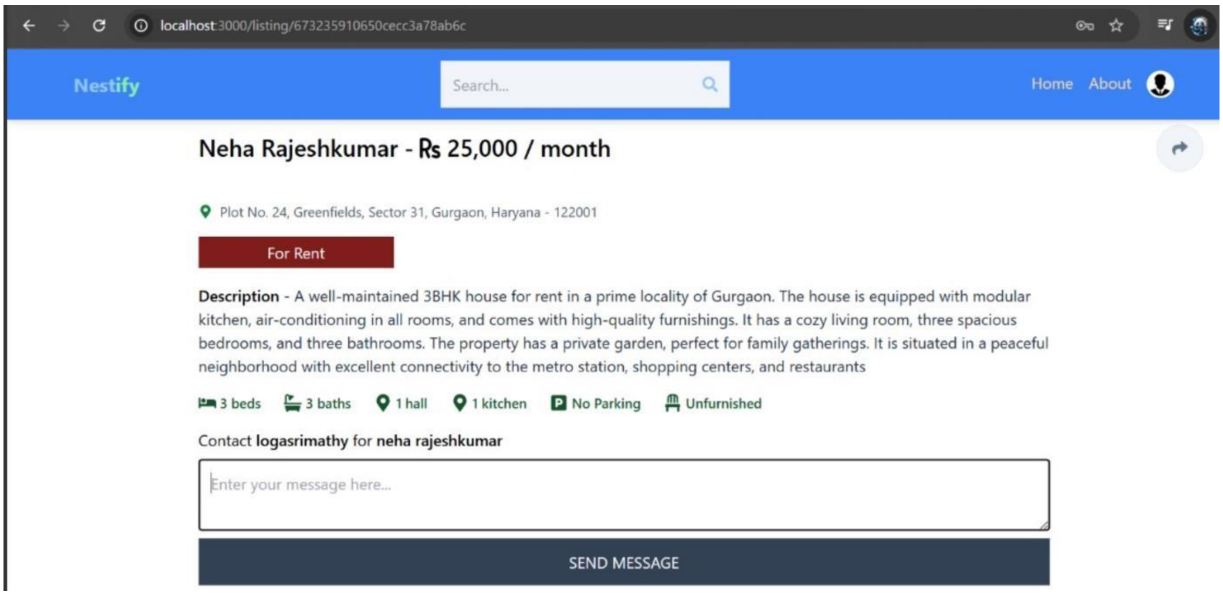
15, MG Road, Sector 12, Vashi, Navi Mumbai,...

A spacious 2BHK flat available for rent in the heart of Vashi, Navi Mumbai. This apartment...

₹20,000 / month

2 BHK

Owner Contact Page:


localhost:3000/listing/673235910650cecc3a78ab6c

Neha Rajeshkumar - ₨ 25,000 / month

Plot No. 24, Greenfields, Sector 31, Gurgaon, Haryana - 122001

For Rent

Description - A well-maintained 3BHK house for rent in a prime locality of Gurgaon. The house is equipped with modular kitchen, air-conditioning in all rooms, and comes with high-quality furnishings. It has a cozy living room, three spacious bedrooms, and three bathrooms. The property has a private garden, perfect for family gatherings. It is situated in a peaceful neighborhood with excellent connectivity to the metro station, shopping centers, and restaurants

3 beds · 3 baths · 1 hall · 1 kitchen · No Parking · Unfurnished

Contact **logasrimathy** for **neha rajeshkumar**

Enter your message here...

SEND MESSAGE

TESTING

To ensure the application functions correctly and meets the requirements, a robust testing strategy is employed. This includes both manual and automated testing methods to validate the functionality, usability, performance, and security of the system.

Testing Strategy

1. **Unit Testing**
   • Focuses on testing individual components or modules of the application (both frontend and backend).
   • Ensures that each unit works as expected in isolation.
   • Example: Testing React components like Login.jsx or API functions in the backend.
2. **Integration Testing**
   • Tests the interaction between different components or modules.
   • Example: Verifying that the frontend communicates correctly with the backend through APIs using Axios.
3. **Functional Testing**
   • Verifies that the system's features work according to the specified requirements.
   • Example: Testing forms to ensure required fields are validated and data is correctly saved.
4. **Regression Testing**
   • Ensures that new updates or changes do not break existing functionality.
   • Example: After implementing a new feature, verify that previously tested workflows remain intact.
5. **Performance Testing**
   • Evaluates the system's speed, responsiveness, and stability under different conditions.
   • Example: Testing the system's ability to handle multiple simultaneous requests.

6. **Security Testing**
   • Ensures that the application is secure and protects user data.
   • Example: Testing user authentication, token validation, and access control to prevent unauthorized access.

Tools Used

Frontend Testing Tools

• Jest: For unit testing React components.
• React Testing Library: For testing user interactions with React components.
• Cypress: For end-to-end testing of the frontend application.

Backend Testing Tools

• Mocha and Chai: For unit and integration testing backend logic and API endpoints.
• Postman: For manual testing of RESTful API endpoints.
• Supertest: For automated API testing in Node.js.

Performance Testing Tools

• JMeter: To simulate high loads and test backend performance under stress.
• Lighthouse: For frontend performance testing.

Security Testing Tools

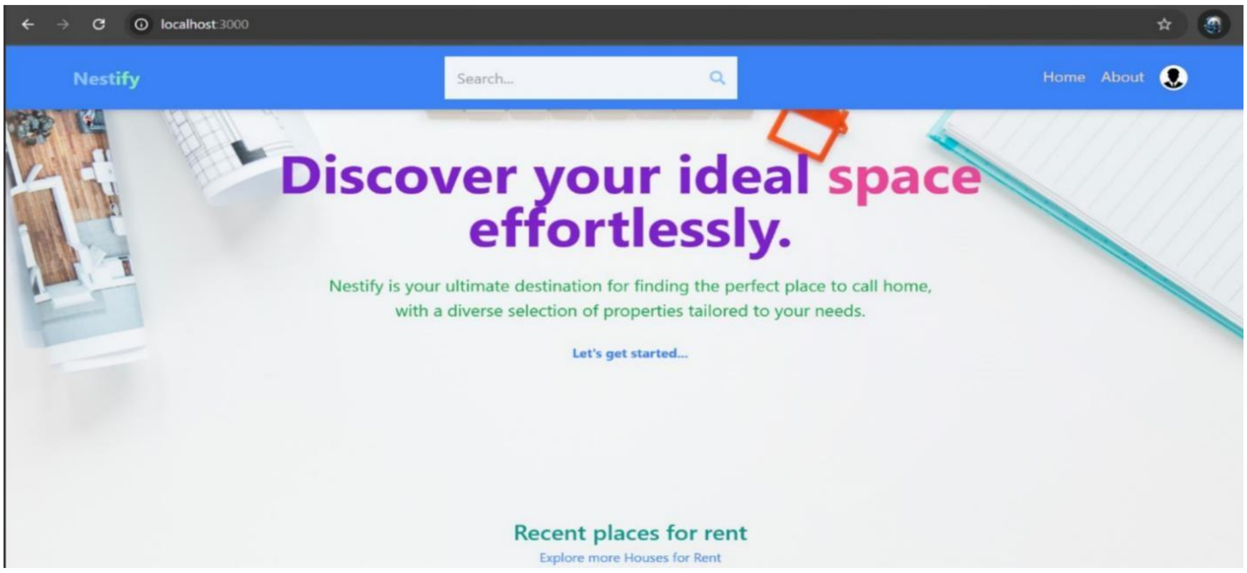
• OWASP ZAP: For automated security testing and identifying vulnerabilities.
• Postman: To check authentication mechanisms and test API security.

Bug Tracking and Reporting Tools

• Trello or Jira: To log and manage bugs identified during testing.
• GitHub Issues: To track and resolve bugs directly within the repository.

SCREENSHOTS

HomePage:



Discover your ideal space effortlessly.

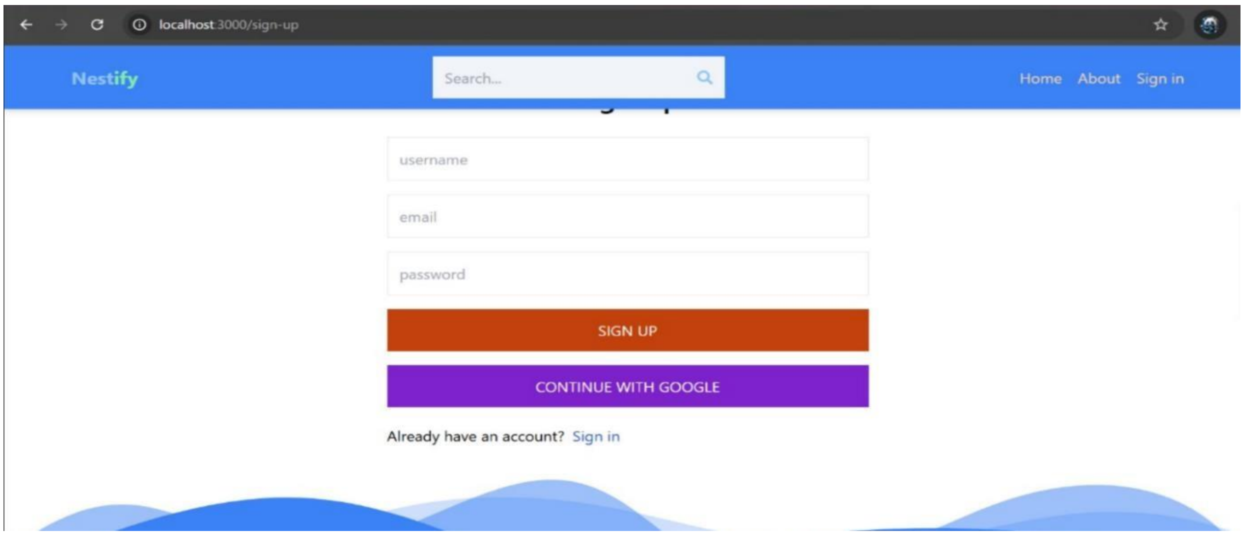
Nestify is your ultimate destination for finding the perfect place to call home, with a diverse selection of properties tailored to your needs.

Let's get started...

Recent places for rent

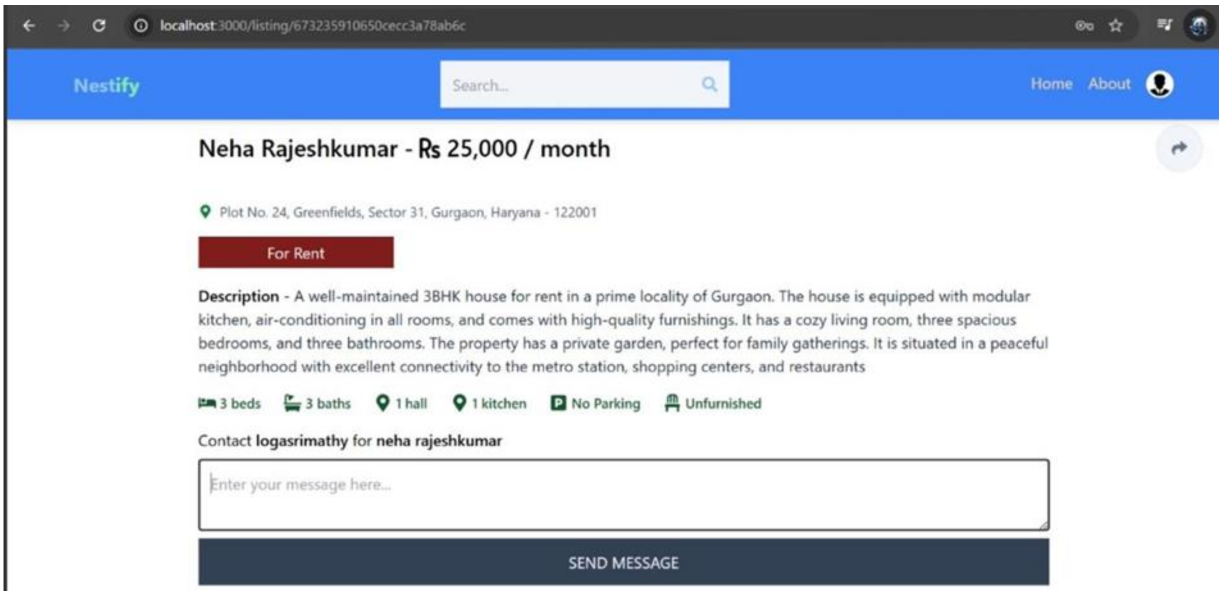
Explore more Houses for Rent

SignUp Page:



Already have an account? Sign in

Owner Contact Page:



localhost:3000/listing/673235910650cecc3a78ab6c

Nestify

Search...

Home About

Neha Rajeshkumar - ₨ 25,000 / month

Plot No. 24, Greenfields, Sector 31, Gurgaon, Haryana - 122001

For Rent

Description - A well-maintained 3BHK house for rent in a prime locality of Gurgaon. The house is equipped with modular kitchen, air-conditioning in all rooms, and comes with high-quality furnishings. It has a cozy living room, three spacious bedrooms, and three bathrooms. The property has a private garden, perfect for family gatherings. It is situated in a peaceful neighborhood with excellent connectivity to the metro station, shopping centers, and restaurants

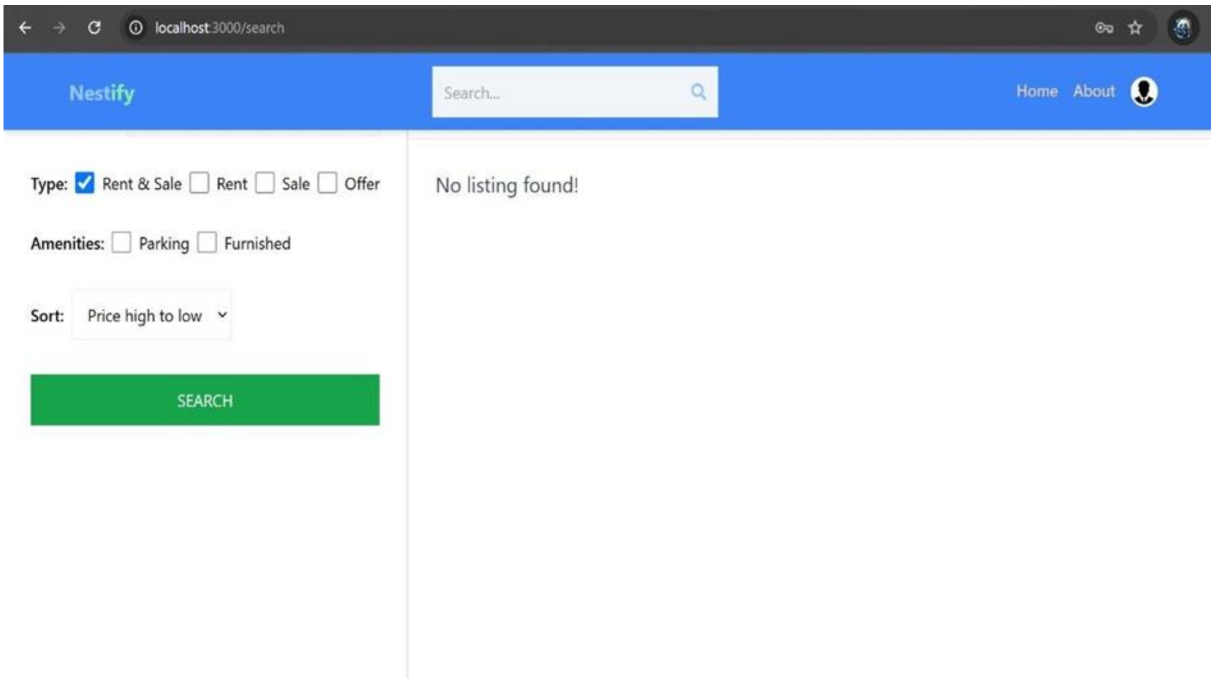
3 beds 3 baths 1 hall 1 kitchen No Parking Unfurnished

Contact **logasrimathy** for **neha rajeshkumar**

Enter your message here...

SEND MESSAGE

Search Page:



localhost:3000/search

Nestify

Search...

Home About

Type: [✓ Rent & Sale] Rent Sale Offer

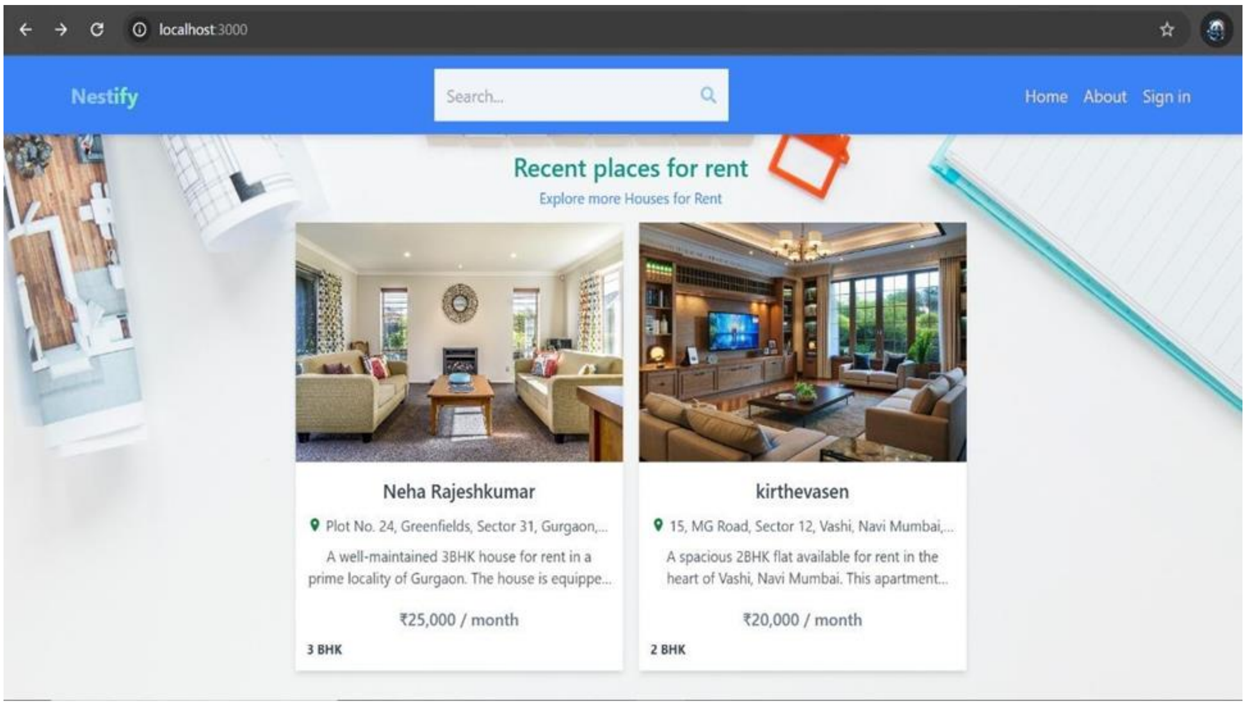
Amenities: Parking Furnished

Sort: Price high to low

SEARCH

No listing found!

Property Details Page:



Recent places for rent

Explore more Houses for Rent

Neha Rajeshkumar

Plot No. 24, Greenfields, Sector 31, Gurgaon,...

A well-maintained 3BHK house for rent in a prime locality of Gurgaon. The house is equippe...

₹25,000 / month

3 BHK

kirthevasen

15, MG Road, Sector 12, Vashi, Navi Mumbai,...

A spacious 2BHK flat available for rent in the heart of Vashi, Navi Mumbai. This apartment...

₹20,000 / month

2 BHK

KNOWN ISSUES

1. Slowdowns Due to High Traffic

• **Issue:** High traffic may cause slowdowns if the backend is not optimized.

• **Impact:** As traffic increases, server response times may degrade, leading to delays in processing requests and longer load times for users.

• **Cause:** Insufficient backend optimization, such as lack of proper indexing, inefficient database queries, or unoptimized server configurations.

• **Workaround:** Load balancing and optimizing database queries, along with server-side caching, can help mitigate these issues. Backend performance improvements are planned for future updates.

• **Status:** To be addressed in the next update.

2. Potential Delays in Email Feature

• **Issue:** Email feature may experience delays under high load.

• **Impact:** Users may experience delays in receiving confirmation or system-generated emails. This can cause frustration, particularly when time-sensitive actions are required.

• **Cause:** The email service may not scale efficiently under high server load, or the system may be sending a high volume of emails at once.

• **Workaround:** Consider using a dedicated email service provider with better scaling capabilities or introducing email batching during high traffic periods.

• **Status:** Investigation into email service optimization is ongoing.

3. Backend Optimization Plan

• **Issue:** High traffic and backend inefficiencies may cause performance degradation.

• **Impact:** Both user experience and system reliability can suffer during peak traffic periods.

• **Cause:** The backend infrastructure may not be fully optimized for handling large volumes of requests or emails concurrently.

• **Workaround:** Future backend improvements, including database optimization, server load balancing, and enhanced email handling, are planned.

• **Status:** Backend optimization and scalability improvements are scheduled for the next release.

FUTURE ENHANCEMENTS

- **Advanced Reporting and Analytics**

  - **Description**: Add a reporting dashboard for admins to generate detailed analytics.
  - **Features**:
    - Complaint resolution time metrics.
    - Department or agent performance analytics.
  - **Benefit**: Helps administrators monitor system efficiency and identify areas for improvement.

- **AI-Powered Complaint Categorization**
  - **Description**: Use machine learning to categorize complaints automatically based on the description.
  - **Features**:
    - Automatic tagging and routing of complaints to the relevant department.
    - Predictive analysis to suggest resolutions based on similar complaints.
  - **Benefit**: Speeds up the complaint assignment process and reduces manual workload.

- **Integration with Social Media and Messaging Platforms**
  - **Description**: Allow users to submit complaints through social media or messaging apps.
  - **Features**:
    - Integration with WhatsApp, Facebook Messenger, or Twitter.
    - Automated replies and status updates via these platforms.
  - **Benefit**: Enhances user convenience and engagement.

- **Mobile App Development**
  - **Description**: Develop native mobile applications for Android and iOS.
  - **Features**:
    - Push notifications for updates and reminders.
    - Offline mode for complaint drafting.
  - **Benefit**: Improves accessibility and enhances the user experience for mobile users.

- **Bulk Actions for Admins**
  - **Description**: Enable administrators to perform bulk actions for managing complaints.
  - **Features**:
    - Bulk complaint assignment.
    - Batch status updates (e.g., mark multiple complaints as resolved).
  - **Benefit**: Increases efficiency in managing large volumes of complaints.

- **Two-Factor Authentication (2FA)**
  - **Description**: Enhance security for user accounts by adding 2FA during login.
  - **Features**:
    - Email or SMS-based OTP for login.
    - Support for authenticator apps like Google Authenticator.
  - **Benefit**: Protects against unauthorized access

- **Real-Time Collaboration Tools**

- **Description**: Add tools for agents to collaborate on complex complaints.
- **Features**:
  - Shared notes on complaints.
  - Internal chat or call feature for agents and admins.
- **Benefit**: Improves coordination and resolution efficiency.

- **Enhanced Notification System**

  - **Description**: Upgrade the notification system for better user communication.
  - **Features**:
    - In-app notifications for all actions.
    - Scheduled reminders for pending complaints.
  - **Benefit**: Keeps users informed and reduces delays in resolution.

- **Customizable User Roles**

  - **Description**: Allow admins to create and assign custom user roles.
  - **Features**:
    - Define permissions for each role.
    - Support for roles beyond admin, agent, and user.
  - **Benefit**: Provides flexibility for organizational needs.

- **Gamification for Agents**

  - **Description**: Introduce gamification to motivate agents and improve performance.
  - **Features**:
    - Points or badges for resolving complaints quickly.
    - Leaderboards to encourage healthy competition.
  - **Benefit**: Boosts morale and enhances productivity.

- **API for Third-Party Integration**

  - **Description**: Develop a public API to allow third-party platforms to integrate with the system.
  - **Features**:
    - Access to complaint submission and tracking features.
    - Secure endpoints for third-party developers.
  - **Benefit**: Extends the usability of the platform beyond the core system.

- **Video or Voice Call Support**

  - **Description**: Enable users to directly connect with agents through video or voice calls.
  - **Features**:
    - Integration with WebRTC for real-time calls.
    - Call recording for future reference.
  - **Benefit**: Improves communication and simplifies complex complaint resolution.

- **Dark Mode**

  - **Description**: Add a dark mode option for the user interface.
  - **Benefit**: Provides a visually comfortable experience, especially for users working at night.

- **Blockchain for Complaint Transparency**

  - **Description**: Use blockchain technology to ensure the immutability and transparency of complaints.
  - **Features**:
    - Store complaint histories on a blockchain ledger.
    - Allow users to verify the integrity of their complaints.
  - **Benefit**: Builds trust and ensures transparency in the complaint resolution process.

- **Online Booking System**

  - **Description**: Add a feature for users to book appointments or services online.
  - **Features**:
    - Real-time availability checking.
    - Automatic booking confirmation and reminders.
  - **Benefit**: Streamlines service scheduling and reduces wait times.

- **Integrated Payment Gateway**

  - **Description**: Enable users to make secure payments directly through the system.
  - **Features**:
    - Support for multiple payment methods (credit card, UPI, PayPal, etc.).
    - Real-time payment status updates and transaction history.
  - **Benefit**: Simplifies payment processes and enhances user convenience.

- **Real-Time Updates and Notifications**

  - **Description**: Provide real-time updates for user actions and system changes.
  - **Features**:
    - Live complaint status tracking.
    - Push notifications for important updates.
  - **Benefit**: Improves user engagement and keeps users informed instantly.

CONCLUSION

The **House Rent App** has successfully streamlined the process of renting properties, offering an efficient solution for both tenants and landlords. Built using the MERN stack (MongoDB, Express.js, React, and Node.js), the app provides a user-friendly platform where tenants can easily search for properties and landlords can manage their listings with ease. The app is designed with scalability in mind, making it suitable for growing user bases and expanding property inventories.

While the app addresses core functionalities well, there are a few areas for improvement, including search accuracy, listing management, and user authentication, which will be addressed in future updates. There are also plans to enhance the property search filters and add features such as analytics for landlords to better track property performance.

The app is positioned to become a vital tool for property management, bridging the gap between landlords and tenants and improving the rental process. As the app evolves, further improvements and features will continue to enhance the user experience, providing a solid foundation for future growth and success.