**Full Stack Development with MERN**

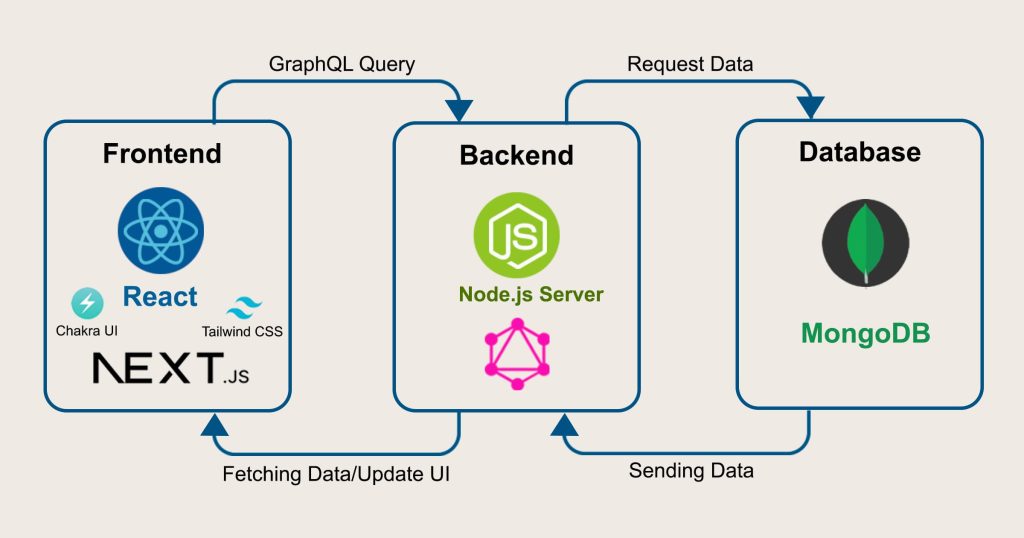
**1. Introduction**

* **Project Title**: House Rent Application Using MERN
* **Team Members**:
  + Bala Ganeshan R- Frontend
  + **Jai Prasanth G** - Backend
  + **Karthick E**- Frontend
  + **Keerthi Vasan M** - Backend

**2. Project Overview**

* **Purpose**:  
  This project aims to streamline the house rental process, providing an efficient and user-friendly platform for tenants and landlords. The platform offers advanced search, secure payment options, and real-time communication, simplifying property management and enhancing the rental experience.
* **Features**:
  + Responsive, user-friendly interface
  + Secure user authentication (JWT-based)
  + Review and rating system for properties and landlords
  + Admin panel for user and content management
  + Email feature to contact the owner.

**3. Architecture**

* **Frontend**:  
  Built with React.js, the frontend offers a responsive UI for users to browse properties, manage bookings, and handle real-time notifications. Core features include a property search page, booking management forms, and user login/signup functionality.
* **Backend**:  
  Developed with Node.js and Express.js, the backend provides the core functionality for user authentication, booking management, property listings, and payment processing. API endpoints support CRUD operations and integrate JWT for secure access.
* **Database**:  
  MongoDB is used as the database to manage user profiles, property listings, bookings,. Mongoose is implemented to define schemas, enforce data validation, and manage interactions with MongoDB collections.  
    
    
    
    
    
  

**4. Setup Instructions**

* **Prerequisites**:
  + Node.js
  + MongoDB
  + npm or yarn
* **Installation**:
  + Clone the repository: git clone <repository-url>
  + Navigate to the client and server directories.
  + Install dependencies:
    - **Frontend**: npm install in the client directory
    - **Backend**: npm install in the server directory
  + Configure environment variables in .env files (e.g., MongoDB URI, JWT secret).

**5. Folder Structure**

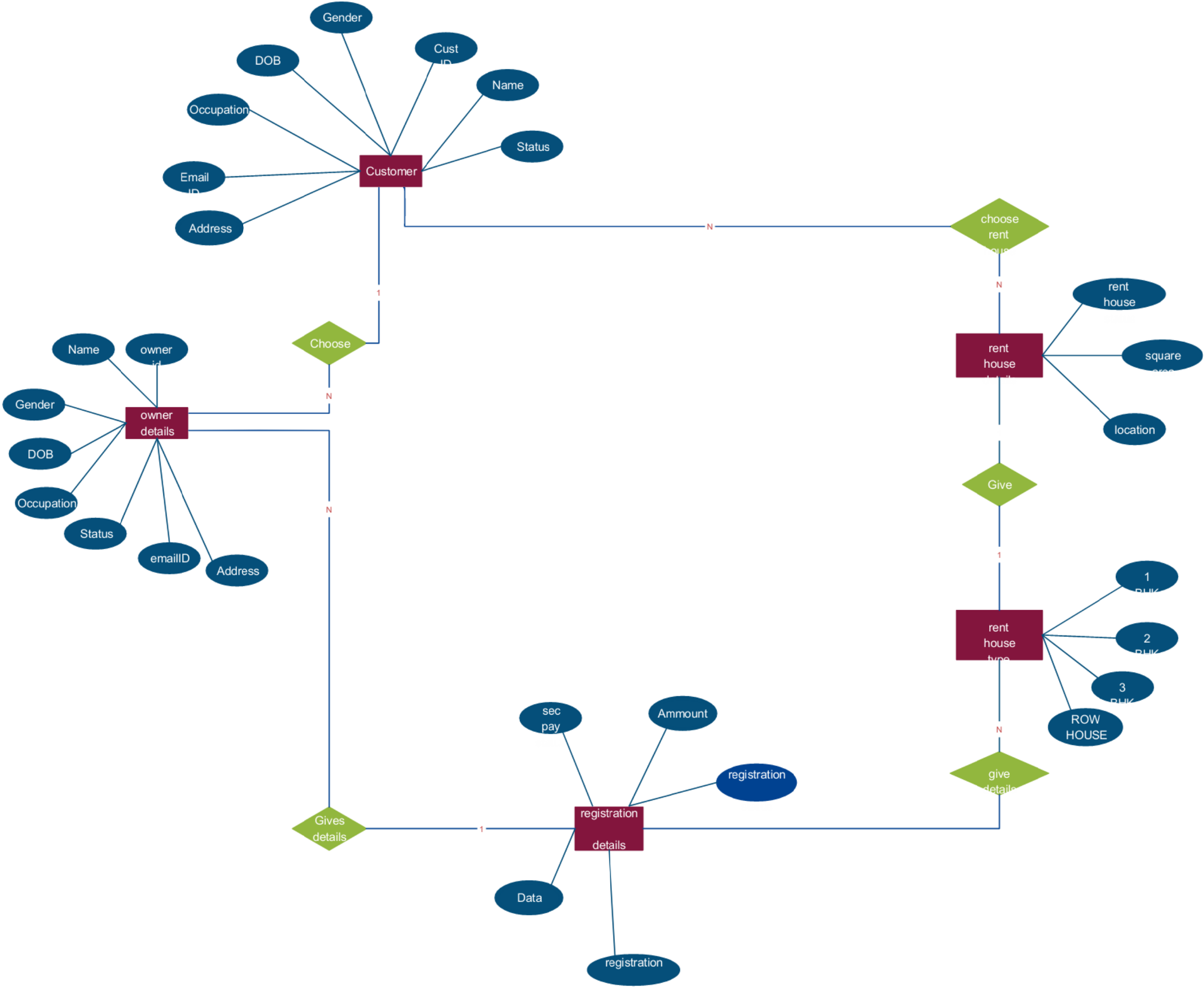
* **Client**:
  + src: Contains the main React components, pages, and styles.
  + components: Includes reusable components like Navbar, Search, PropertyCard, etc.
  + pages: Contains main pages such as Home, Property Details, and Booking.
  + services: Handles API requests and state management for different features.
* **Server**:
  + controllers: Manages logic for user authentication, property, booking, and payment operations.
  + models: Defines schemas for users, properties, bookings, and payments.
  + routes: Organizes API endpoints, such as /api/users, /api/properties, and /api/bookings.
  + config: Contains configuration for MongoDB and JWT.

**6. Running the Application**

* **Frontend**:  
  Start the frontend server in the client directory: npm start
* **Backend**:  
  Start the backend server in the server directory: npm start

**7. API Documentation**

* **Users**:
  + POST /api/users/register: Registers a new user.
  + POST /api/users/login: Authenticates user and provides a JWT token.
* **Properties**:
  + GET /api/properties: Fetches all properties based on filters.
  + POST /api/properties: Allows landlords to add new properties.



**8. Authentication**

* **Method**:  
  JWT-based authentication. Users receive a JWT token upon login, which is required to access protected routes. Role-based access controls differentiate permissions between tenants and landlords.

**9. User Interface**

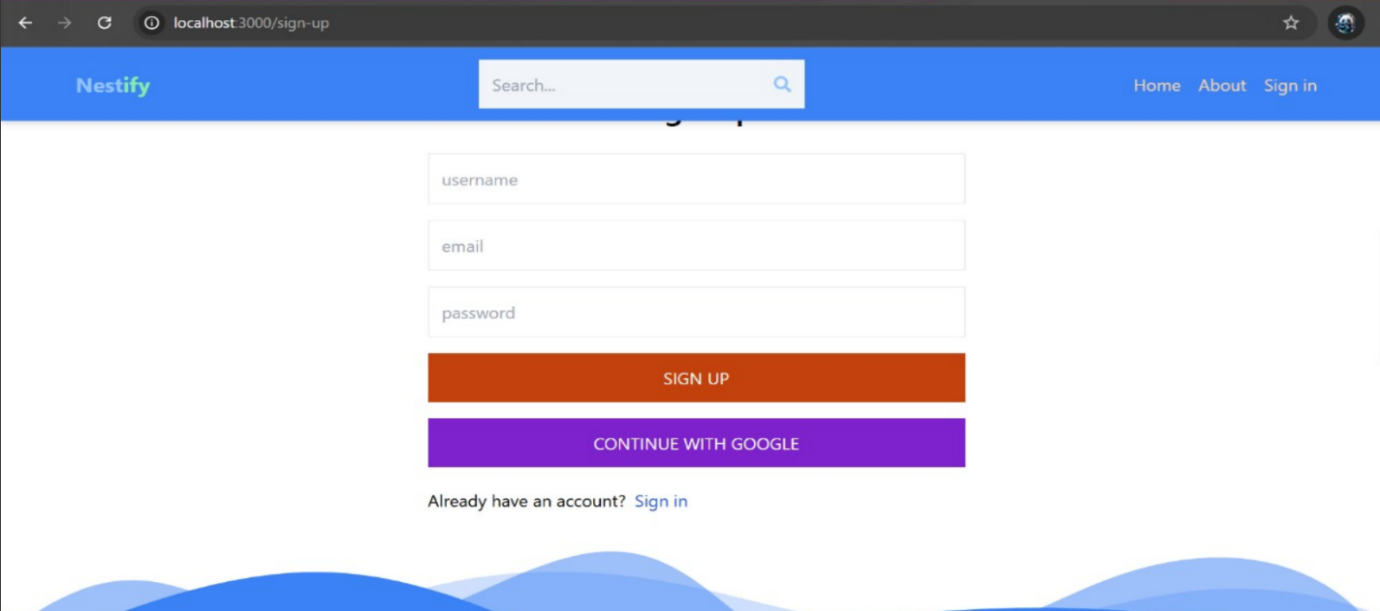
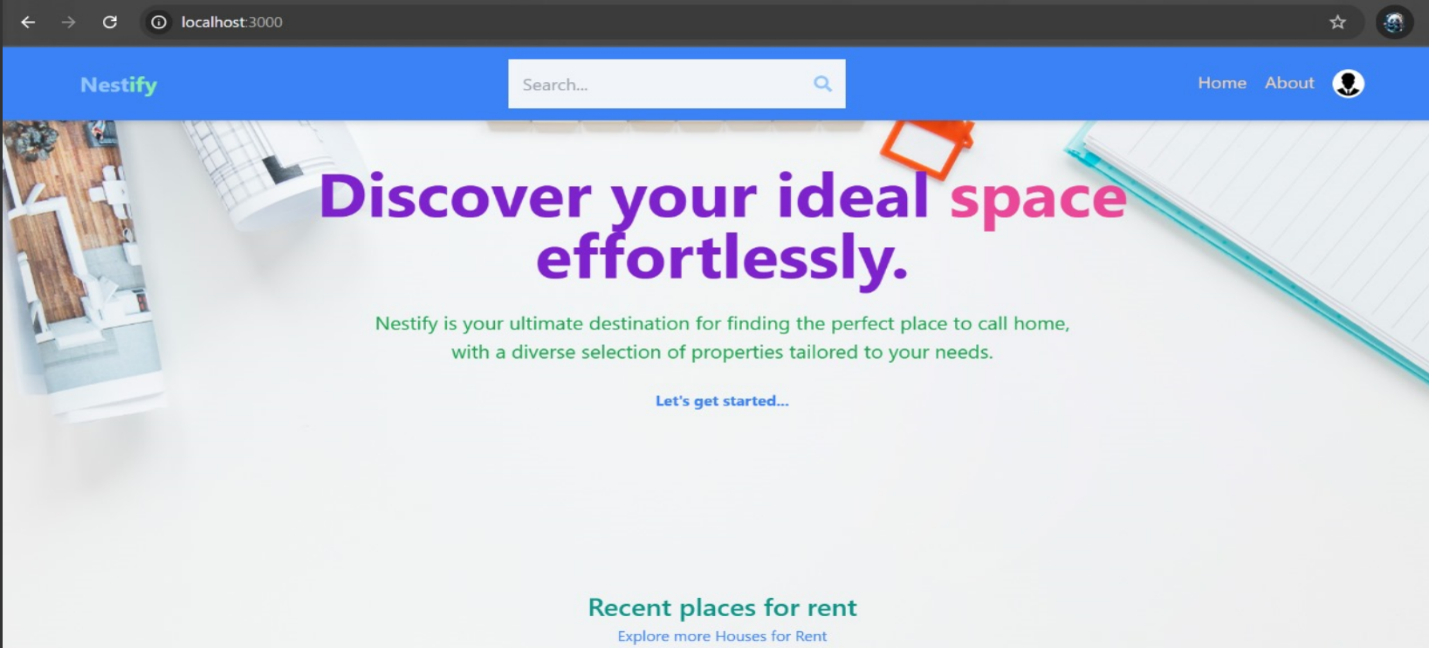
* **Overview**:  
  The UI includes pages like the registration page, home page, login page, property details. These pages are responsive and provide intuitive navigation for both tenants and landlords.

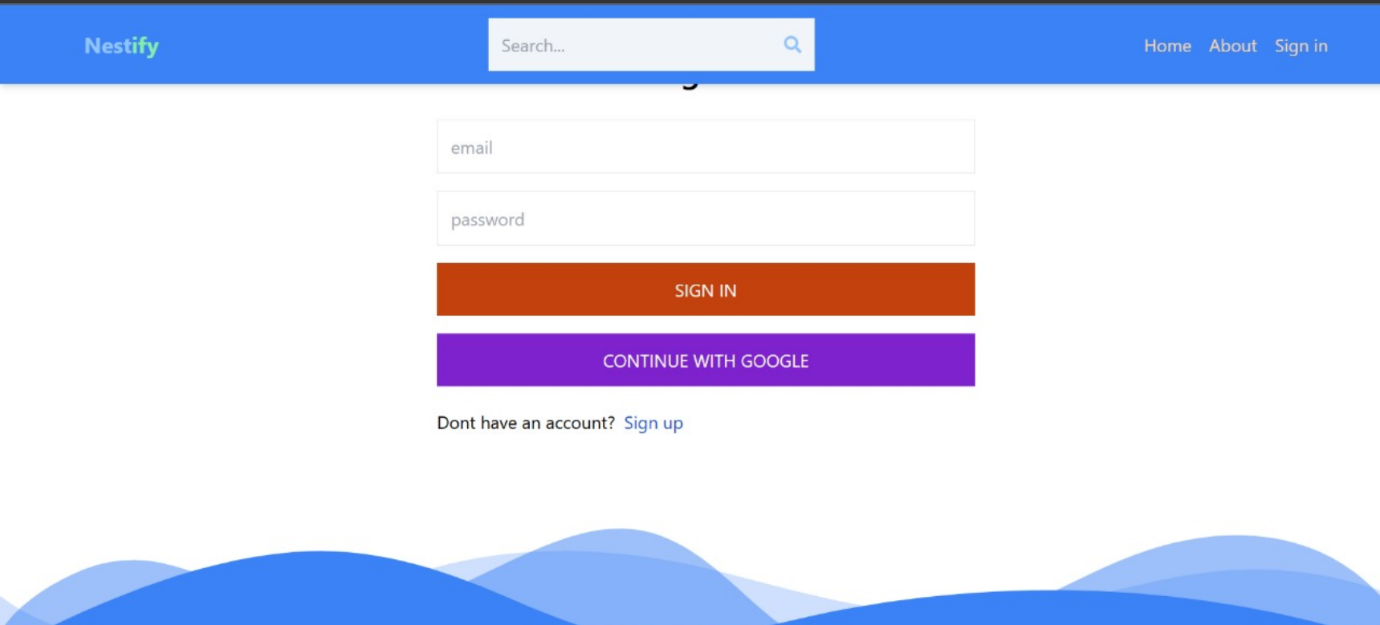
**10. Testing**

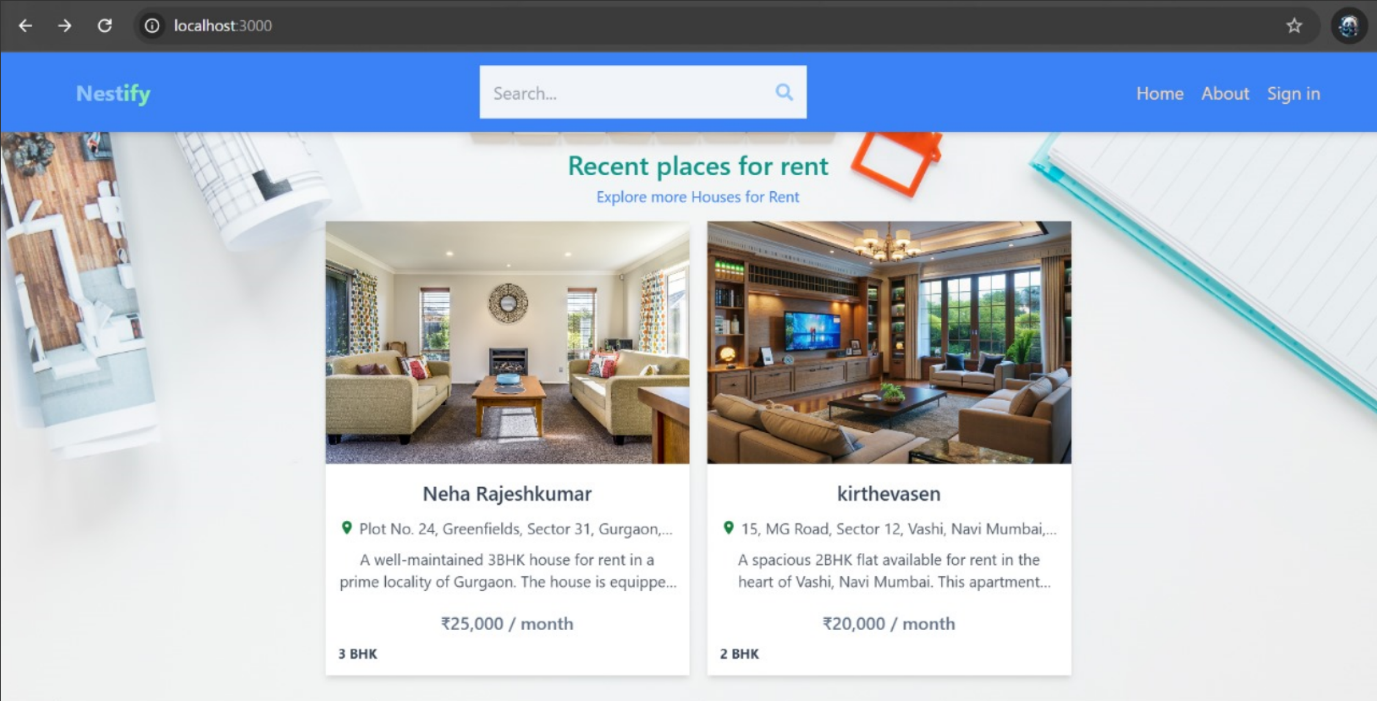
* **Strategy**:
  + Unit testing for individual components using Jest.
  + Integration testing for API endpoints.
  + End-to-end testing for major user flows.

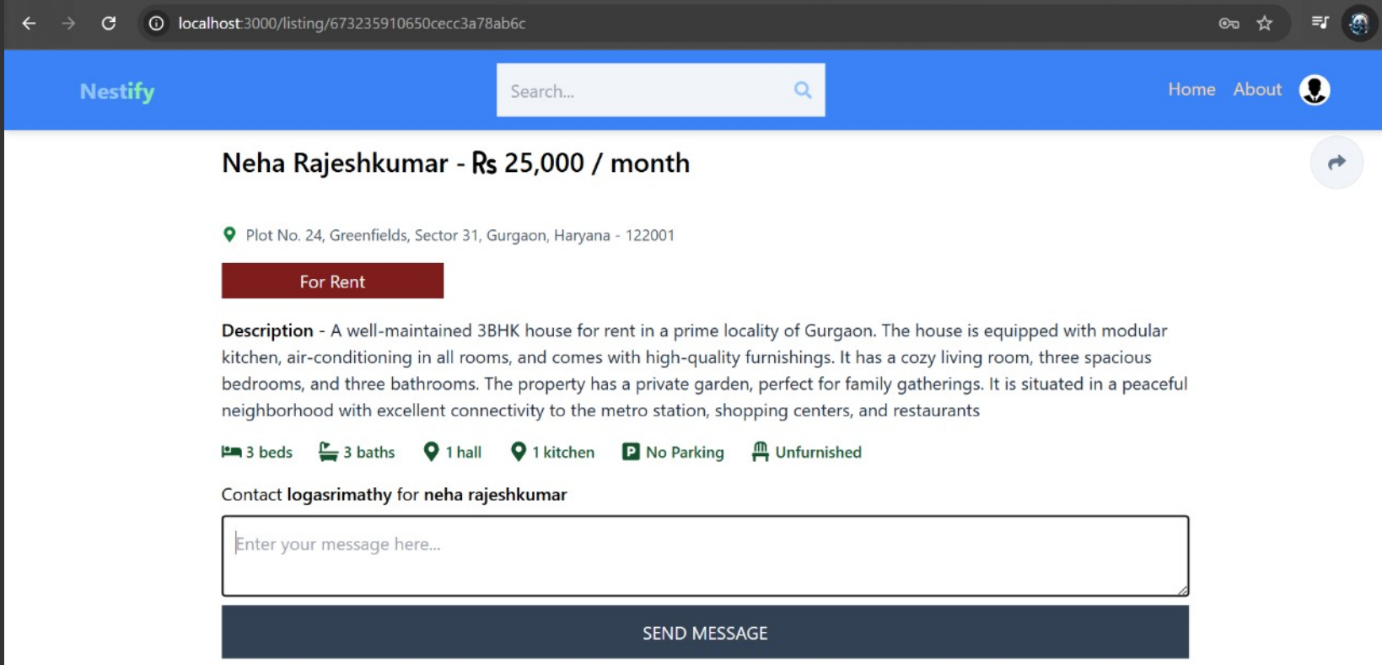
**11. Screenshots or Demo**  
**Demo Link:**

**Screenshots:**  
  
  
**Registration Form:**

  
  
  
 **Home Page**  
  


**Login Page  
  
**

**Property Details**  
  


**Owner Contact:  
  
**

**12. Known Issues**

* High traffic may cause slowdowns if the backend is not optimized.
* Real-time features (e.g., messaging) may experience delays under high load.

**13. Future Enhancements**

* Advanced search and filtering options
* User profile customization
* Mobile app version
* Multilingual support
* Automated booking and payment processing