JEPPIAAR
ENGINEERING COLLEGE

JEPPIAAR NAGAR, RAJIVGANDHI SALAI
CHENNAI – 600119.

DEPARTMENT OF INFORMATION TECHNOLOGY

IV YEAR B. TECH – VII SEM

ACADEMIC YEAR 2024 - 25 (ODD SEM)

NM1042 - MERN Stack Powered by MongoDB
House Rent App Project Report

(Naan Mudhalvan Project)

TEAM MEMBERS

Naveen C	- 310821205054
Keshore Kumar K	- 310821205036
Madheesh R	- 310821205041
Murugesan P	- 310821205050

JEPPIAAR ENGINEERING COLLEGE

JEPPIAAR NAGAR, RAJIVGANDHI SALAI, CHENNAI – 600119.

DEPARTMENT OF
INFORMATION TECHNOLOGY

This is a Bonafide Record Work of ____

Register No. _________________________ submitted for the Anna University Practical

Examination held on ___________________ in **NM1042 - MERN Stack Powered by**

MongoDB during the year ____.

Signature of the Faculty-In-Charge **Signature of the HOD**

Date: __________________ **Examiners Internal:** ______________________

External: ________________________________

COLLEGE VISION & MISSION

Vision

To build Jeppiaar Engineering College as an institution of academic excellence in technological and management education to become a world class university.

Mission

- To excel in teaching and learning, research and innovation by promoting the principles of scientific analysis and creative thinking.
- To participate in the production, development and dissemination of knowledge and interact with national and international communities.
- To equip students with values, ethics and life skills needed to enrich their lives and enable them to contribute for the progress of society.
- To prepare students for higher studies and lifelong learning, enrich them with the practical skills necessary to excel as future professionals and entrepreneurs for the benefit of Nation's economy.

Program Outcomes

PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision

To produce engineers with excellent knowledge in the field of Information Technology through scientific and practical education to succeed in an increasingly complex world.

Mission

- To demonstrate technical and operational excellence through creative and critical thinking for the effective use of emerging technologies.
- To involve in a constructive, team-oriented environment and transfer knowledge to enable global interaction.
- To enrich students with professional integrity and ethical standards that will make them deal social challenges successfully in their life.
- To devise students for higher studies and perpetual learning, upgrade them as competent engineers and entrepreneurs for country's development.

Program Educational Objectives (PEOs)

PEO 1	To support students with substantial knowledge for developing and resolving mathematical, scientific and engineering problems
PEO 2	To provide students with adequate training and opportunities to work as a collaborator with informative and administrative qualities
PEO 3	To shape students with principled values to follow the code of ethics in social and professional life
PEO 4	To motivate students for extensive learning to prepare them for graduate studies, R&D and competitive exams
PEO 5	To cater the students with industrial exposure in an endeavor to succeed in the emerging cutting-edge technologies

Program Specific Outcomes

PSO1	Students are able to analyze, design, implement and test any software with the programming and testing skills they have acquired.
PSO2	Students are able to design algorithms, data management to meet desired needs, for real time problems through analytical, logical and problem solving skills.
PSO3	Students are able to provide security solutions for network components and data storage & management which will enable them to work in the industry ethically.

Course Outcomes (COs)

C407.1	Configure various virtualization tools such as Virtual Box, VMware workstation
C407.2	Design and deploy a web application in a PaaS environment
C407.3	Learn how to simulate a cloud environment to implement new schedulers.
C407.4	Install and use a generic cloud environment that can be used as a private cloud.
C407.5	Install and use Hadoop

Table of Contents

1. Introduction

Project Title: House Rent App (MERN Stack)

Team Members and Roles:

- **Naveen C [310821205054]**: *Team Lead and Backend Developer* - Responsible for API development, database integration, and deployment.
- **Keshore Kumar K [310821205036]:** *Frontend Developer* - Designed the user interface, implemented React components, and ensured responsive design.
- **Madheesh R [310821205041]**: *Database Manager* - Designed and managed MongoDB schemas, ensured data consistency, and optimized queries.
- **Murugesan P [310821205050]**: *Quality Assurance and Documentation* - Conducted application testing, bug fixes, and prepared the documentation.

2. Project Overview

- **Purpose**: Simplifies finding and renting properties by connecting landlords and tenants via a modern, user-friendly platform.
- **Technology:** Developed using the MERN Stack - MongoDB, Express.js, React.js, and Node.js.
- **Key Features:**
  - User Authentication: Secure account management for landlords and tenants.
  - Property Search & Listings:
  - Intuitive search with filters for location, price, and property type.
  - Landlords can upload property details with images and descriptions.
  - Tenants can browse listings, save favorites, and contact landlords via integrated messaging.
- **Enhanced Functionalities:**
  - Payment processing for rental transactions.
  - Reviews and ratings for properties.
  - Dashboard for managing rental agreements.

3. Architecture

Frontend:

The frontend is built using **React.js**. Key libraries and tools include:

- **React Router**: For managing routes and navigation.
- **Redux**: To handle application state effectively.
- **Axios**: For making API calls to the backend.

Backend:

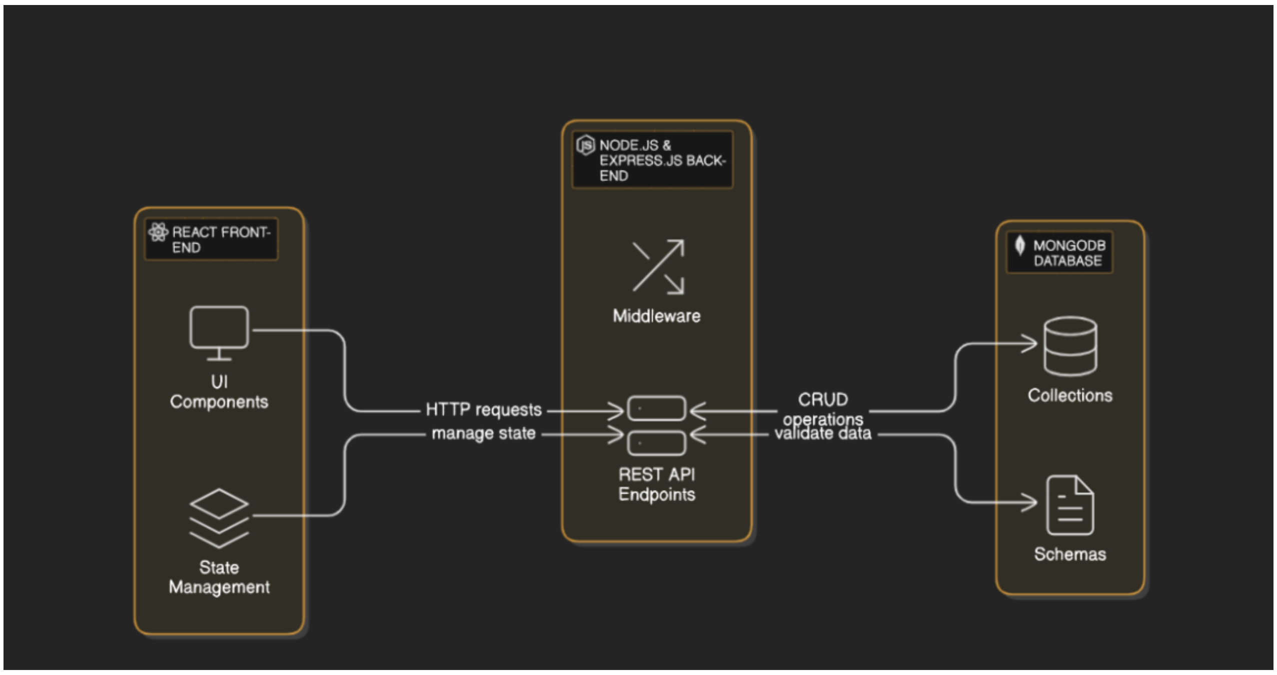
The backend uses **Node.js** and **Express.js**. It serves as the intermediary between the frontend and the database. Key highlights include:

- RESTful API design.
- Middleware for authentication, error handling, and request validation.
- Modular architecture with separate controllers, routes, and middleware.

Database:

The database is **MongoDB**, storing collections such as:

- **Users**: Stores user credentials and profiles.
- **Listings**: Includes house rent details, availability, and pricing.
- **Bookings**: Records user bookings, payment details, and status.



4. Setup Instructions

Essential Software & Tools

1. **Node.js (v14 or higher)**: Install along with npm or Yarn (package managers).
2. **MongoDB**:
    - Local installation or access to a cloud-hosted instance (e.g., MongoDB Atlas).
3. **React**:
    - Ensure compatibility with related packages like react-router-dom (for routing) and axios (for API requests).

Back-End Dependencies

- **Express.js**: Framework for building APIs.
- **Mongoose**: MongoDB object modeling.
- **Additional Libraries**:
    - dotenv (environment variable management).
    - cors (handle cross-origin requests).
    - jsonwebtoken and bcryptjs (for authentication and password encryption).

Development Tools

- **Code Editor**: Visual Studio Code (preferred).
- **Version Control**: Git for managing codebase.
- **API Testing**: Postman (for REST API testing).
- **Development Utilities**: nodemon (for automatic server restarts).

INSTALLATION

1. **Clone the Repository**:
    - Download or clone the project repository from its source (e.g., GitHub).
2. **Install Prerequisites**:
    - Ensure **Node.js** and **MongoDB** are installed on your system.
3. **Navigate to the Project Folder**:
    - Locate the server (backend) and client (frontend) directories inside the project folder.
4. **Install Dependencies**:
    - Run npm install in both the server and client directories to install the required packages.
5. **Set Up the Database**:
    - Start the MongoDB server locally or connect to a cloud-hosted database (e.g., MongoDB Atlas).
    - Update the database connection string in the .env file located in the server directory.

6. **Start the Backend Server**:
   - In the server directory, run npm start or node index.js.
7. **Start the Frontend**:
   - In the client directory, run npm start.
8. **Access the Application**:
   - Open a browser and visit http://localhost:3000 to use the app.
9. **Environment-Specific Adjustments**:
   - Update any necessary settings in .env or configuration files for full functionality.

5. Folder Structure

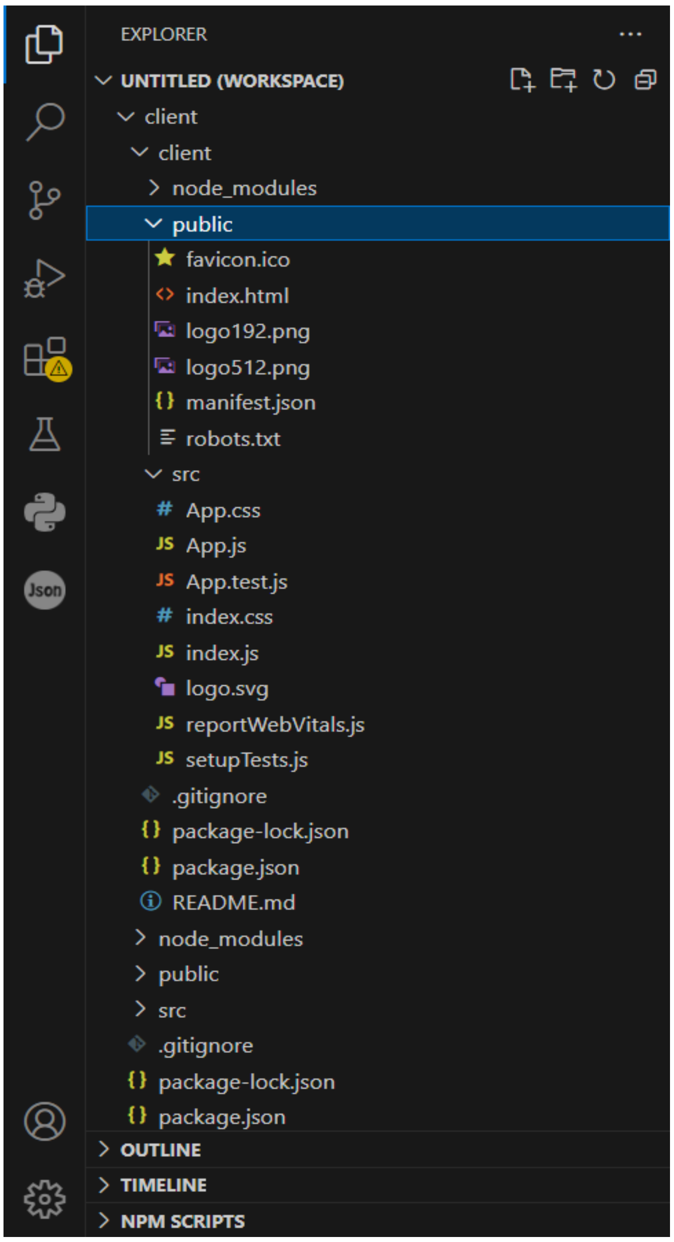
Client (React):

```
client/
├── src/
│   ├── components/   # Reusable UI components
│   ├── pages/        # Route-specific pages
│   ├── redux/        # State management logic
│   └── App.js        # Main application entry point
└── public/           # Static assets
```

Server (Node.js):

```
server/
├── models/           # MongoDB schemas
├── routes/           # AflI routes
├── controllers/      # Business logic
├── middleware/       # Request
validation/authentication
└── index.js          # Entry point of the backend
server
```

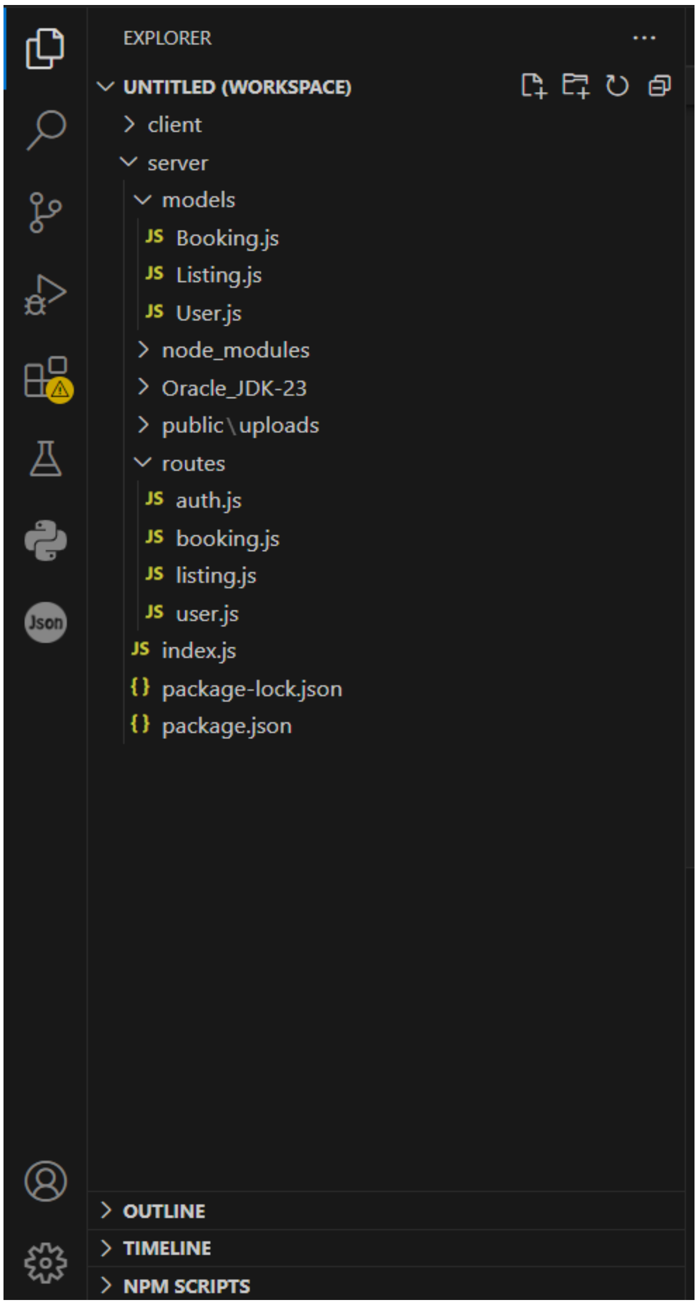
CLIENT


EXPLORER
UNTITLED (WORKSPACE)

```
client
  client
    node_modules
    public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
    src
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
  node_modules
  public
  src
  .gitignore
  package-lock.json
  package.json
```

OUTLINE
TIMELINE
NPM SCRIPTS

SERVER



```
EXPLORER
UNTITLED (WORKSPACE)
  > client
  v server
    v models
        Booking.js
        Listing.js
        User.js
    > node_modules
    > Oracle_JDK-23
    > public\uploads
    v routes
        auth.js
        booking.js
        listing.js
        user.js
      index.js
      package-lock.json
      package.json
> OUTLINE
> TIMELINE
> NPM SCRIPTS
```

6. Running the Application

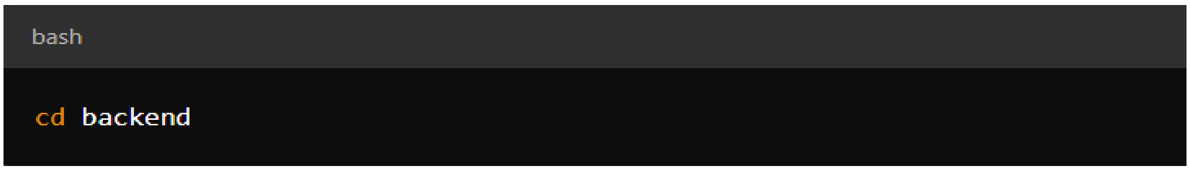
To run the frontend and backend servers of a MERN stack application locally for a house rent app, follow these steps. Make sure you have Node.js, npm, and MongoDB installed on your system.

BACKEND

- **Start the Backend Server**

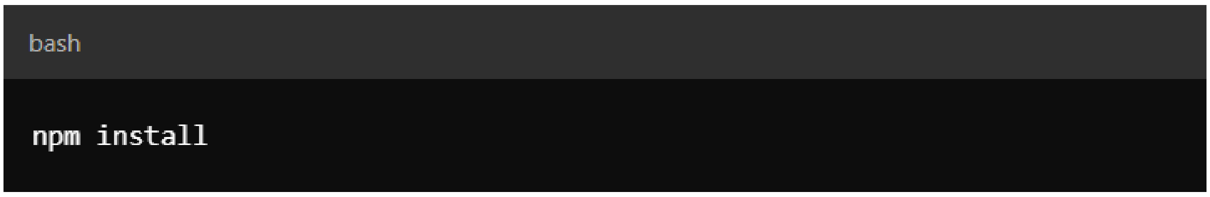
  The backend typically uses Node.js and Express.

  Navigate to the backend directory:



```bash
cd backend
```

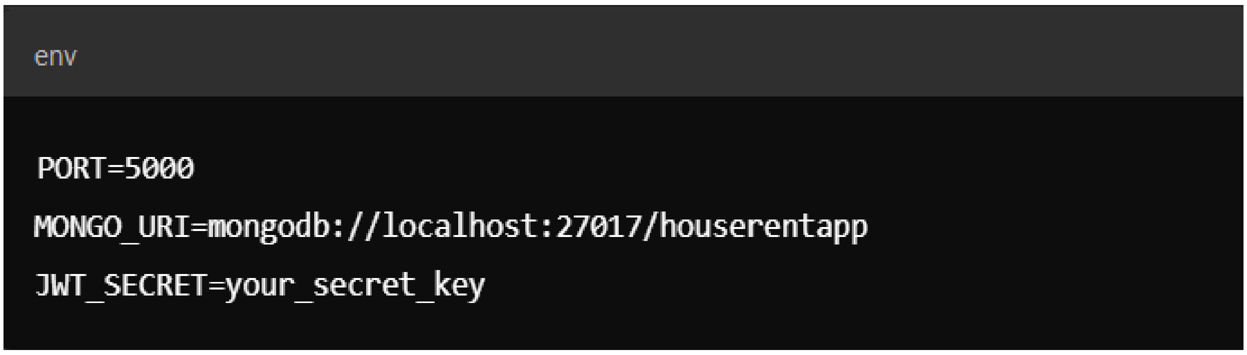
2.Install dependencies:



```bash
npm install
```

3.Set up environment variables:

- Create a .env file in the backend directory.
- Add necessary configurations, such as



```env
PORT=5000
MONGO_URI=mongodb://localhost:27017/houserentapp
JWT_SECRET=your_secret_key
```

4.Run the server:

```bash
npm start
```

or if you are using Nodemon for auto-reloading:

```bash
npm run dev
```

The backend server will typically start at http://localhost:5000.

FRONTEND

Start the Frontend Server

The frontend typically uses React.

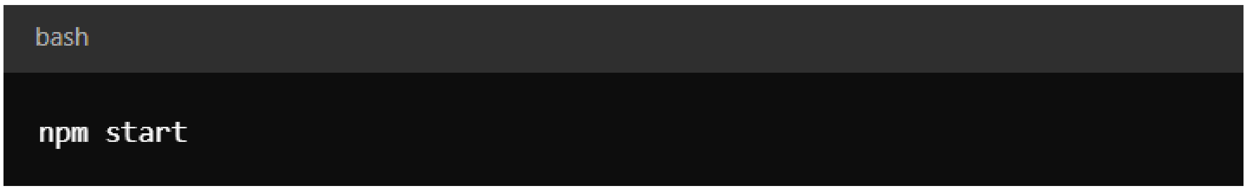
1.Navigate to the frontend directory:

```bash
cd frontend
```

2.Install dependencies:

```bash
npm install
```

3.Start the React development server:



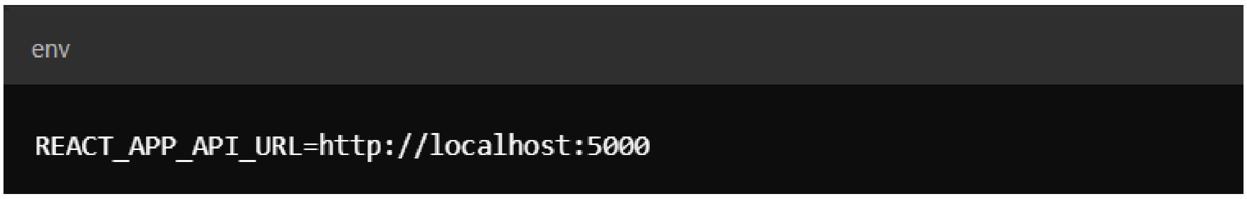
```bash
npm start
```

- The frontend server will typically start at http://localhost:3000.

4. Connect the Frontend and Backend

- Ensure the frontend communicates with the backend by setting the correct API base URL in

the React app. For example, in the frontend configuration (e.g., src/config.js or .env):



```env
REACT_APP_API_URL=http://localhost:5000
```

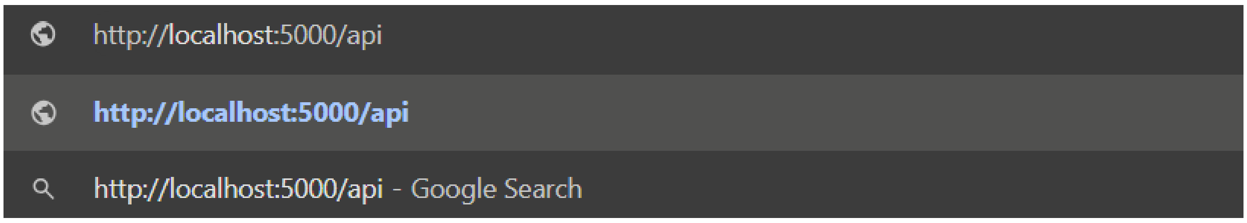
5. Verify the Application

- Open http://localhost:3000 in your browser to access the frontend.
- The frontend should interact with the backend running on http://localhost:5000.This setup assumes default configurations. Adjust paths, environment variables, or scripts based on your specific project structure.

7. API Documentation

Creating API documentation for a house rent app using the MERN stack involves listing the backend routes, HTTP methods, expected request payloads, and responses. Below is a sample API documentation template. It assumes the app has features like user authentication, property listings, and booking functionalities.

Base URL
http://localhost:5000/api



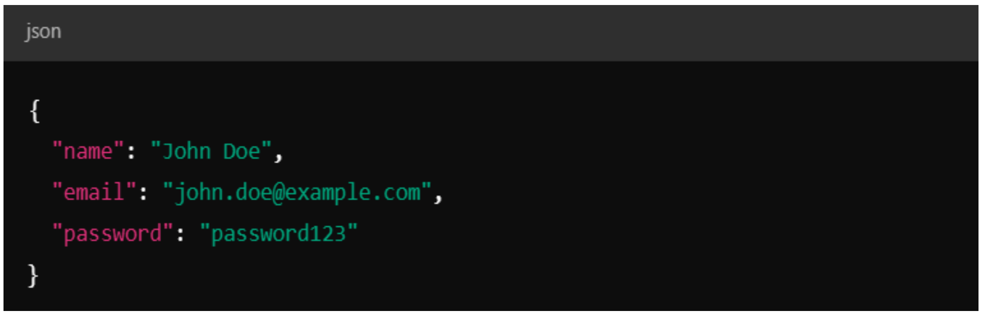
http://localhost:5000/api

http://localhost:5000/api

http://localhost:5000/api - Google Search

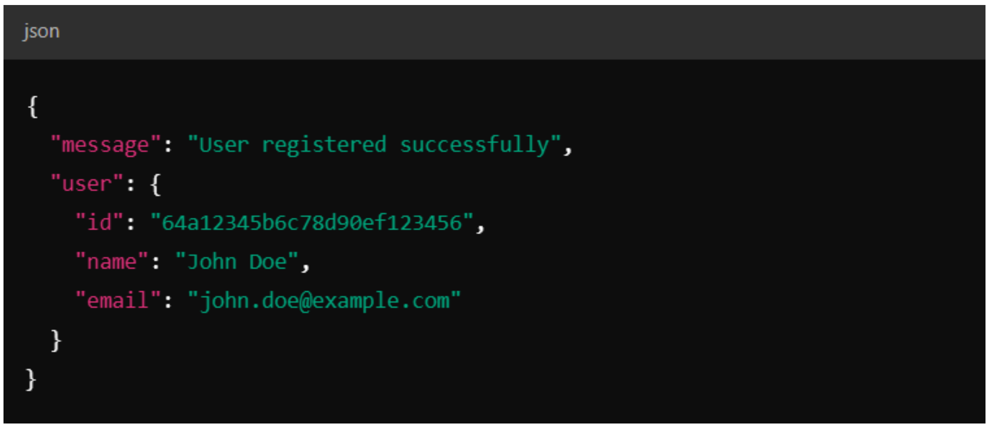
Authentication Routes

1. User Registration
   Endpoint: POST /auth/register
   Description: Register a new user.
   Request Body:



```json
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "password123"
}
```

Response:
201 Created



```json
{
  "message": "User registered successfully",
  "user": {
    "id": "64a12345b6c78d90ef123456",
    "name": "John Doe",
    "email": "john.doe@example.com"
  }
}
```

2. User Login
Endpoint: POST /auth/login
Description: Authenticate a user.
Request Body:

```json
{
  "email": "john.doe@example.com",
  "password": "password123"
}
```

Response:
200 OK

```json
{
  "message": "Login successful",
  "token": "jwt-token"
}
```

Property Routes

1.Get All Properties
 Endpoint: GET /properties
 Description: Retrieve all available properties.

2. Get Property by ID
Endpoint: GET /properties/:id
Description: Retrieve details of a single property.

3.Add a New Property
Endpoint: POST /properties
Description: Add a new property (for logged-in users).

4.Update a Property
Endpoint: PUT /properties/:id
Description: Update property details (for the owner).

5.Delete a Property
Endpoint: DELETE /properties/:id
Description: Delete a property (for the owner).

8. Authentication

Authentication ensures that a user is who they claim to be. In a MERN application, this is implemented using **JWT (JSON Web Tokens)** and a combination of Node.js, Express.js, and MongoDB

1.User Registration:

- Users register by providing their details (e.g., name, email, and password).
- Passwords are hashed using bcrypt before storing them in the database to ensure security.
- Example flow:User submits a POST request to /auth/register.
- Backend:Hashes the password using bcrypt.
- Saves the user data in MongoDB.
- Responds with a success message.

2.User Login:

- Registered users log in by submitting their email and password.
- The backend verifies the email exists and compares the password hash with the database record.
- Upon successful login, a JWT is generated and sent to the user.
- Example flow:User submits a POST request to /auth/login.
- Backend:Validates credentials.
- Generates a JWT using a secret key and sends it in the response.

AUTHORIZATION

Authorization determines what actions an authenticated user can perform. In the house rent app, this could mean restricting access to specific resources or operations.

1. Role-Based Access

- Roles: Typically, two roles are implemented:
- User: Can search properties, book rentals, and view their bookings.
- Admin (or Property Owner): Can manage (add, update, delete) properties and view all bookings.

2. Protecting Routes

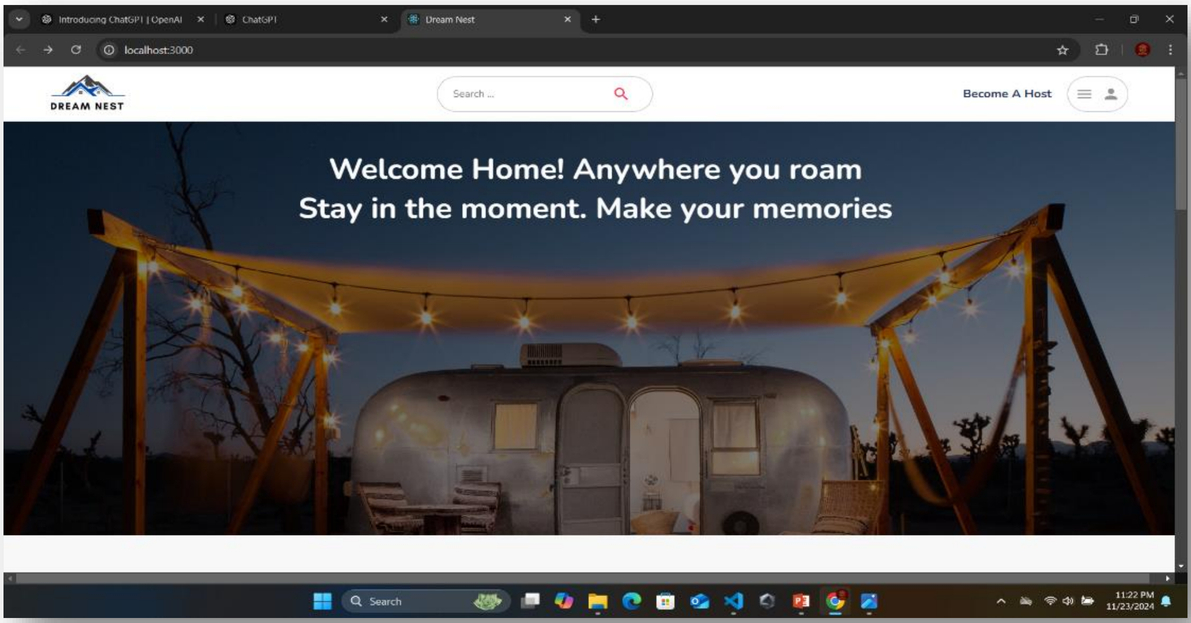
a) Middleware for Authentication:

- Middleware extracts the JWT from the Authorization header and verifies it.
- If valid, the middleware attaches the user’s details to the request object (req.user) and allows access.

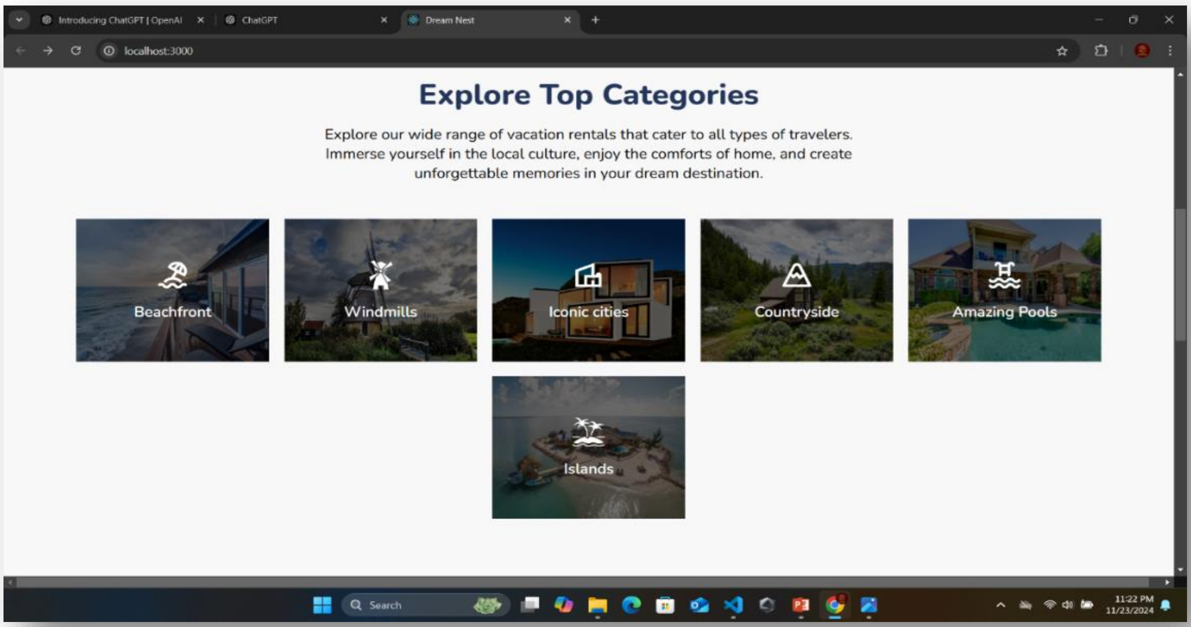
b) Middleware for Authorization:

- Checks the user's role to determine if they are allowed to access the resource.

9. User Interface

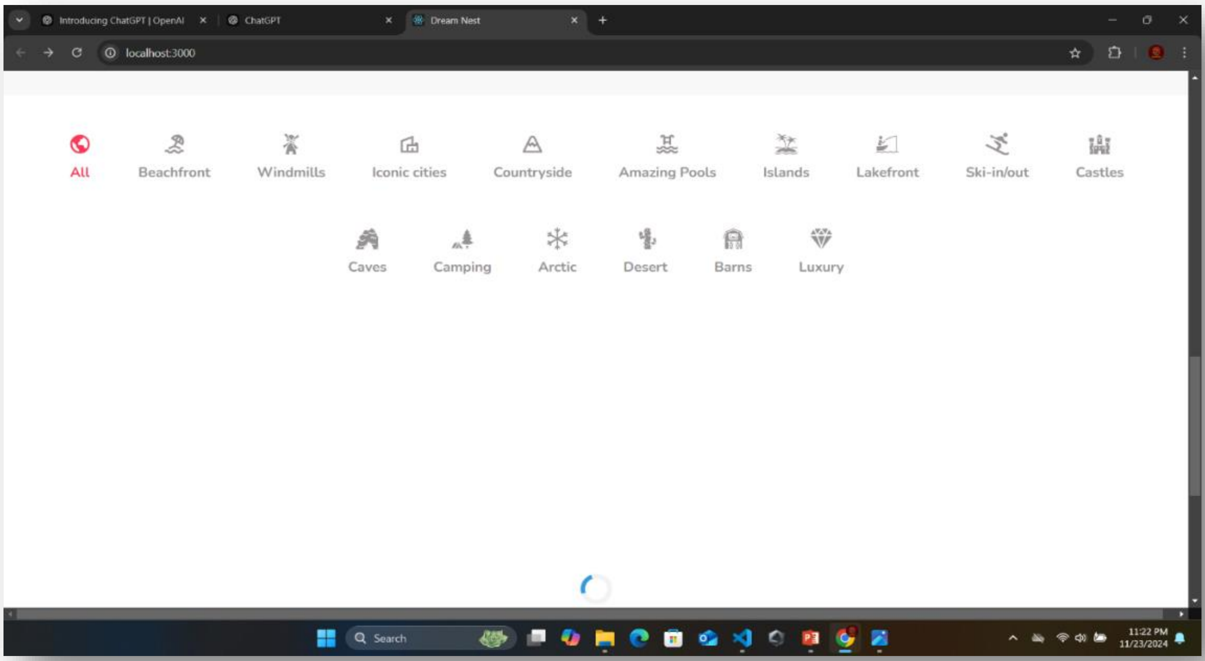


Welcome Home! Anywhere you roam
Stay in the moment. Make your memories



Explore Top Categories

Explore our wide range of vacation rentals that cater to all types of travelers. Immerse yourself in the local culture, enjoy the comforts of home, and create unforgettable memories in your dream destination.



10. Testing

Testing is an essential part of the development lifecycle to ensure the reliability and quality of a MERN (MongoDB, Express.js, React, Node.js) stack application. Below is an overview of the testing strategy and the tools that can be used for testing different components of a house rent application built using the MERN stack.

Types of Testing

The testing strategy for a MERN stack house rent app generally involves several types of testing, including unit testing, integration testing, and end-to-end (E2E) testing. Each type ensures the application works as expected from different perspectives.

a) Unit Testing

Unit testing focuses on testing individual components or functions in isolation to ensure they behave as expected.

- **Backend Unit Testing:** Test each API route, middleware, business logic, and utility function.
- **Frontend Unit Testing:** Test individual components, hooks, and UI logic in React.

b) Integration Testing

- Integration testing checks if multiple components or systems work together as expected. This includes testing how the frontend interacts with the backend and how different parts of the system (e.g., database, API) work together.

c) End-to-End (E2E) Testing

- End-to-end tests verify that the entire application works as expected, from the user interface down to the database. It simulates real user actions like signing up, logging in, and booking a property.

d) Performance and Security Testing

- Performance testing measures how the system performs under load, while security testing checks for vulnerabilities, especially with user authentication and authorization.

Testing Tools

Backend Testing Tools

1.Jest
Purpose: A popular testing framework for Node.js applications, used for unit testing and integration testing.

2. Supertest
Purpose: A library for HTTP assertions, commonly used with Jest to test API endpoints.

3. Chai (with Mocha)
Purpose: A BDD (Behavior-Driven Development) assertion library for Node.js, commonly used with Mocha (a test framework).

b) Frontend Testing Tools

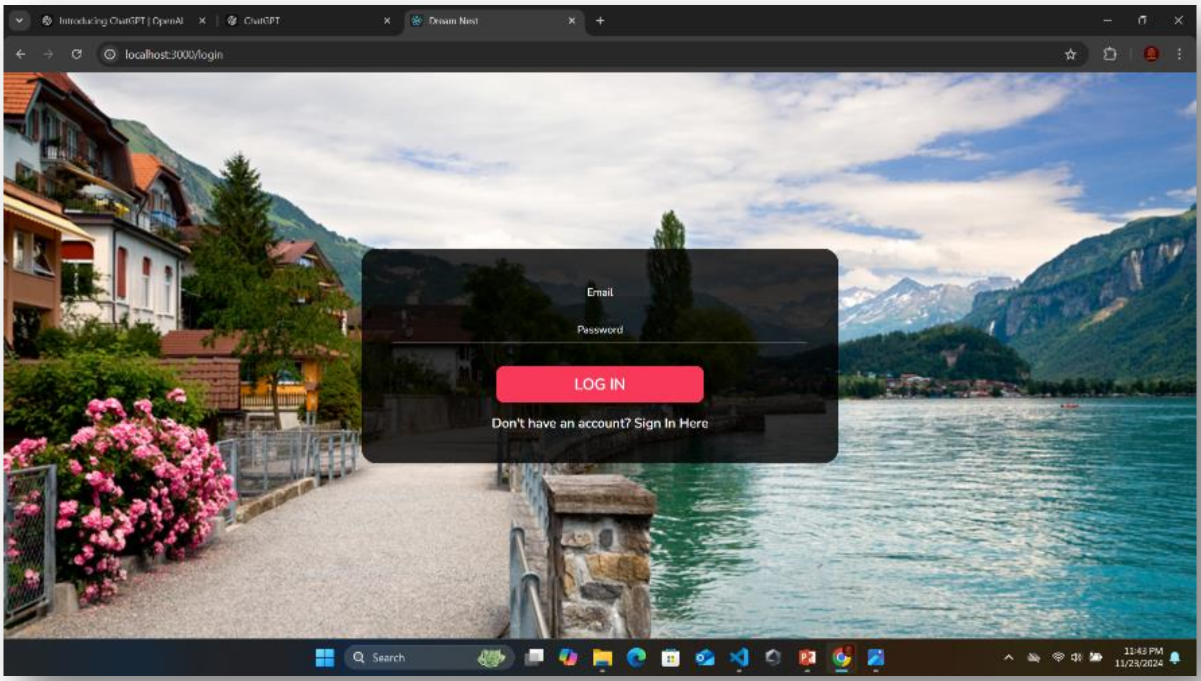
1.Jest (with React Testing Library)
Purpose: Jest is also commonly used for testing React components. React Testing Library encourages testing UI components from the user's perspective.

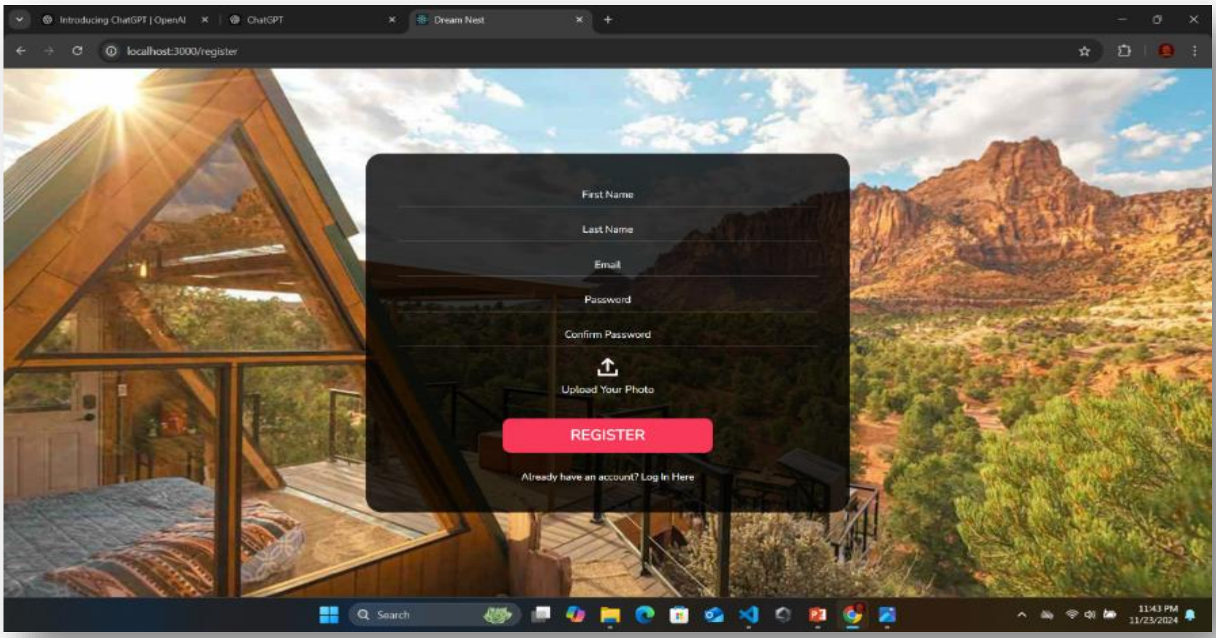
2.Cypress
Purpose: End-to-end testing framework for testing React applications.

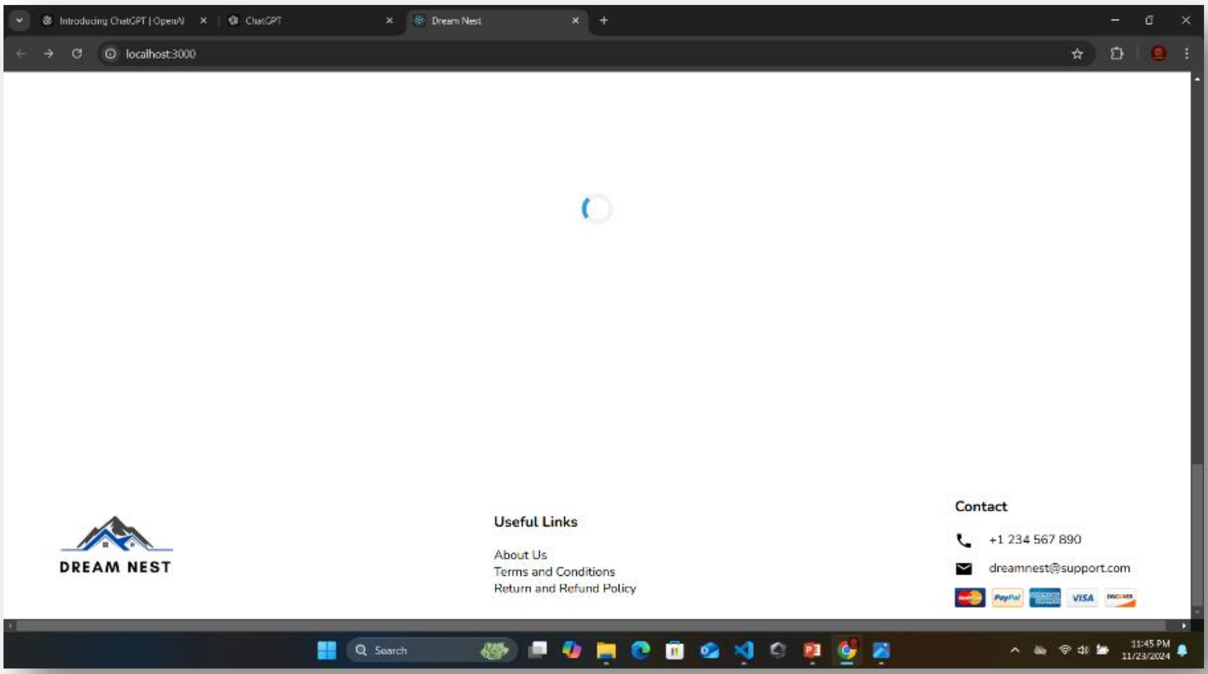
3.Enzyme (Optional)
Purpose: A JavaScript testing utility for React that helps with shallow rendering and manipulating the component tree.

11. Screenshots or Demo

Screenshots:





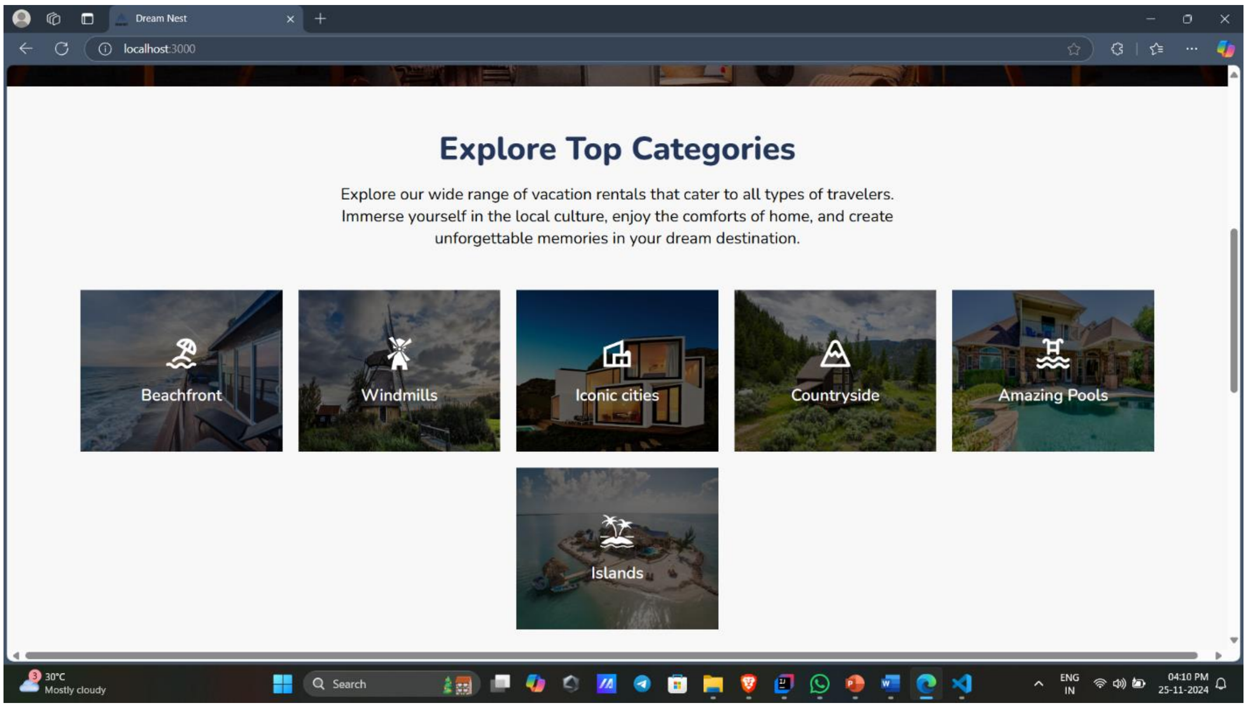


Useful Links

About Us
Terms and Conditions
Return and Refund Policy

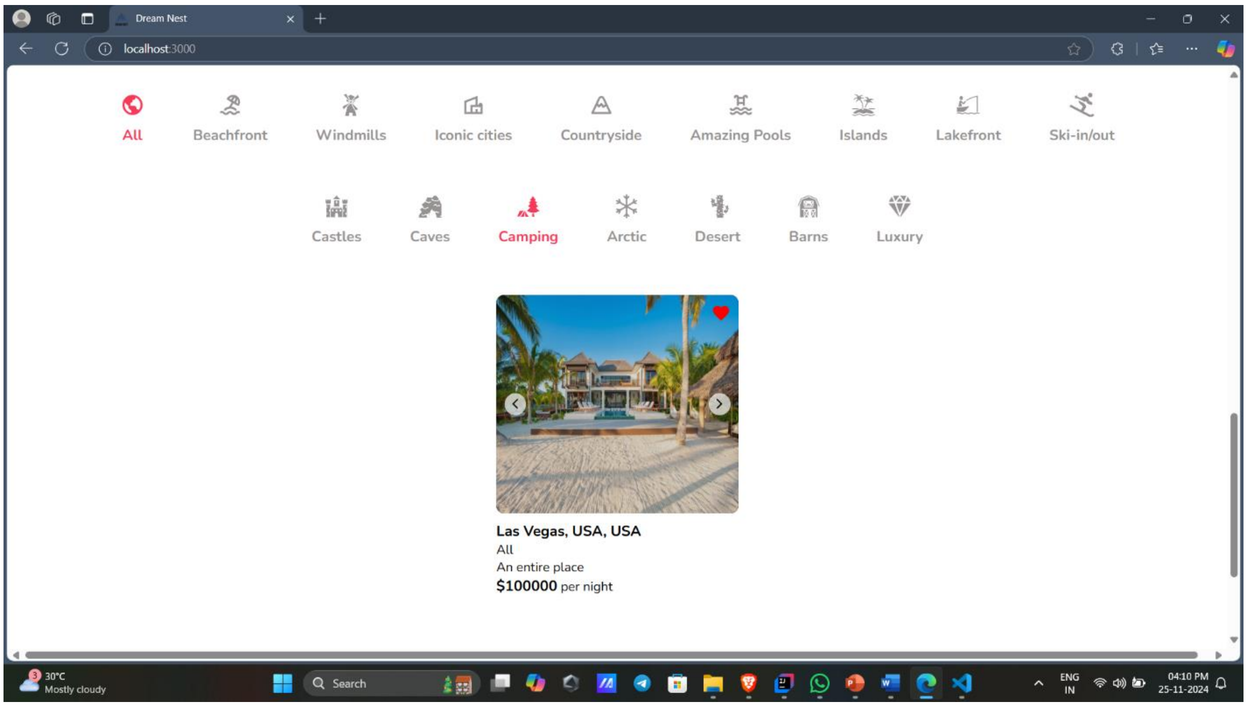
Contact

+1 234 567 890
dreamnest@support.com

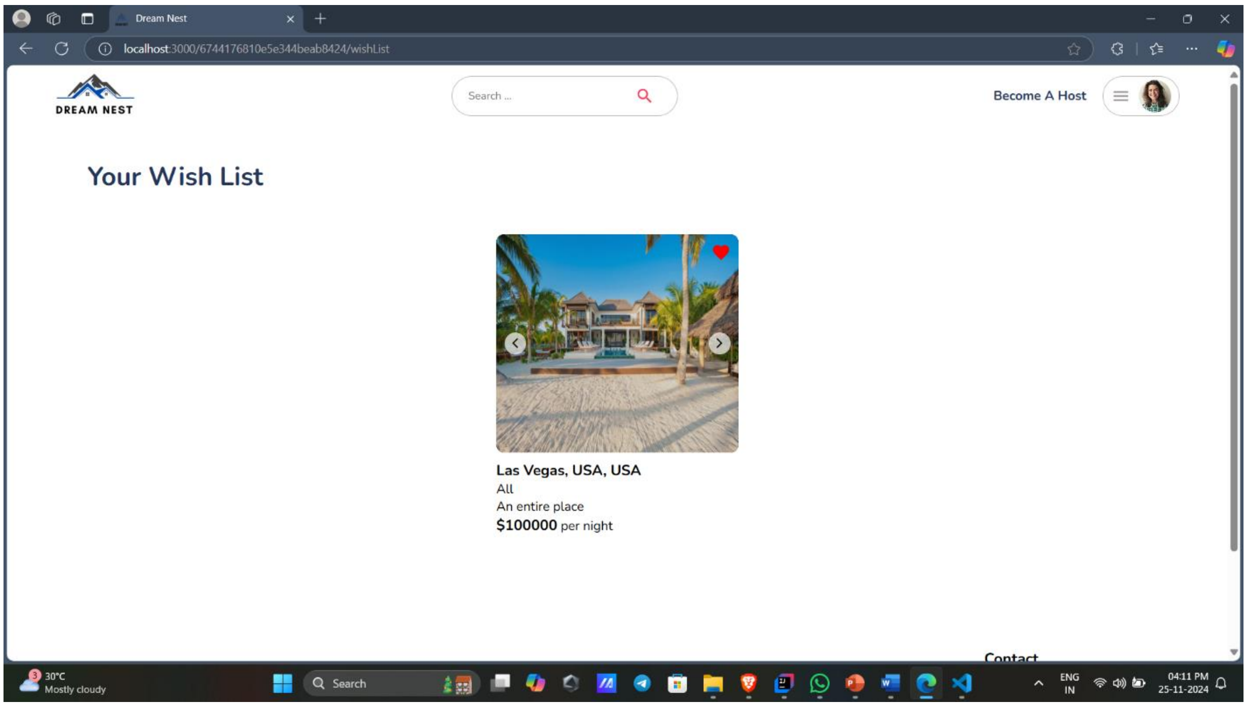


Explore Top Categories

Explore our wide range of vacation rentals that cater to all types of travelers. Immerse yourself in the local culture, enjoy the comforts of home, and create unforgettable memories in your dream destination.

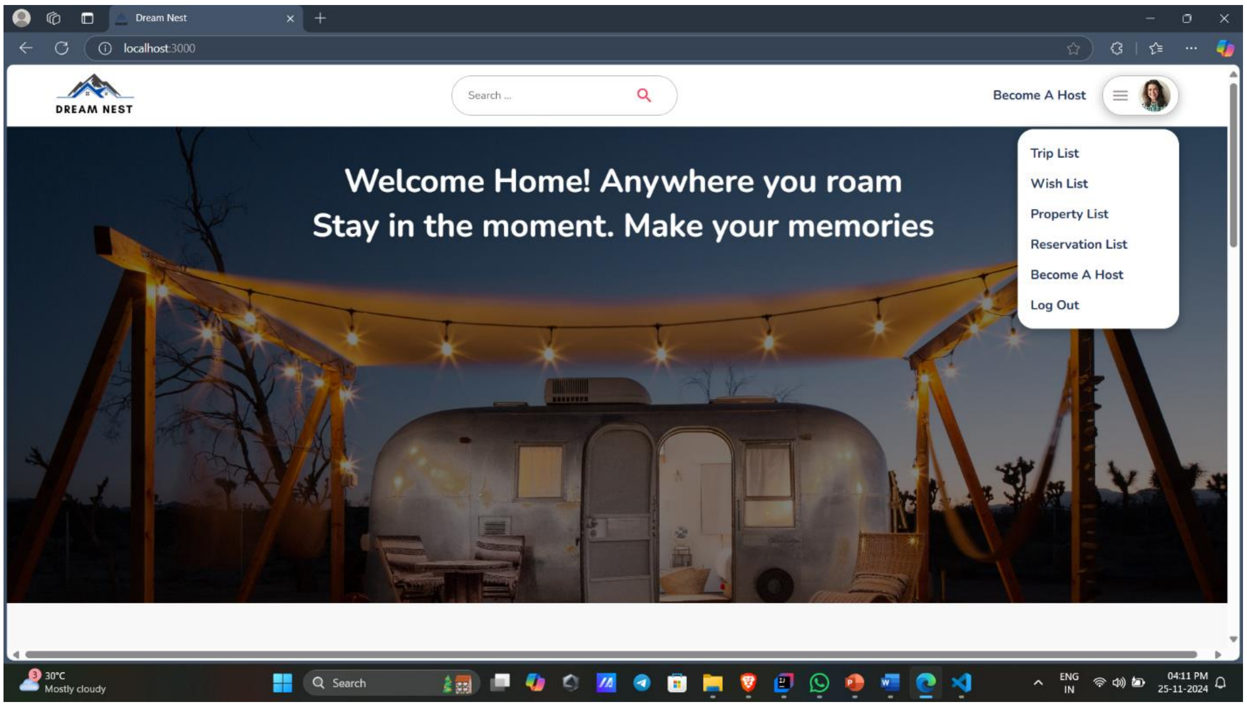


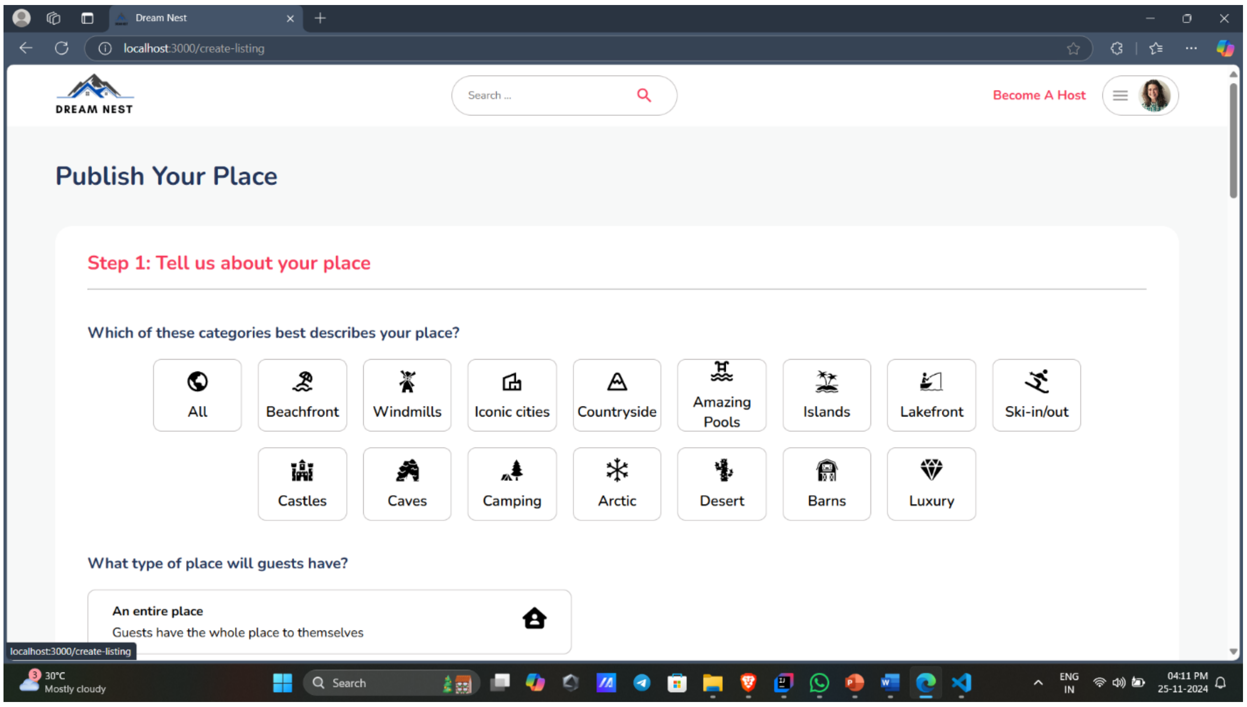
Las Vegas, USA, USA
All
An entire place
$100000 per night



Your Wish List

Las Vegas, USA, USA
All
An entire place
$100000 per night





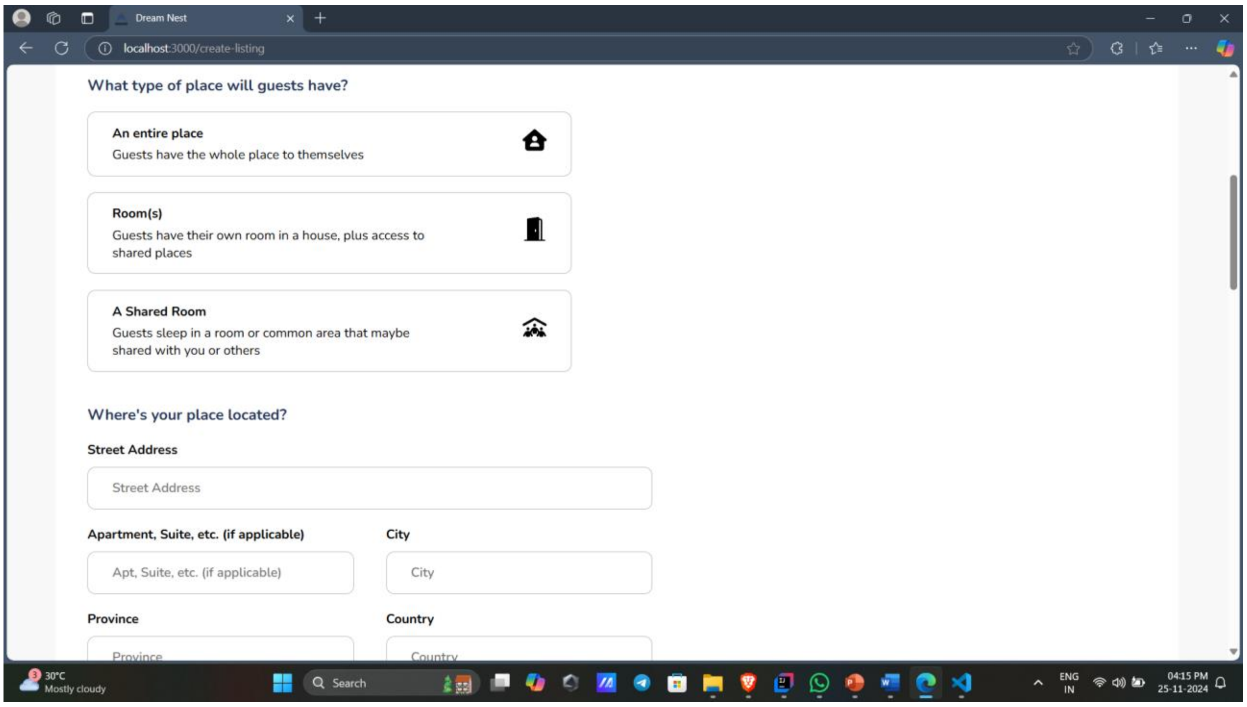
Publish Your Place

Step 1: Tell us about your place

Which of these categories best describes your place?

What type of place will guests have?

An entire place
Guests have the whole place to themselves


What type of place will guests have?

An entire place
Guests have the whole place to themselves

Room(s)
Guests have their own room in a house, plus access to shared places

A Shared Room
Guests sleep in a room or common area that maybe shared with you or others

Where's your place located?

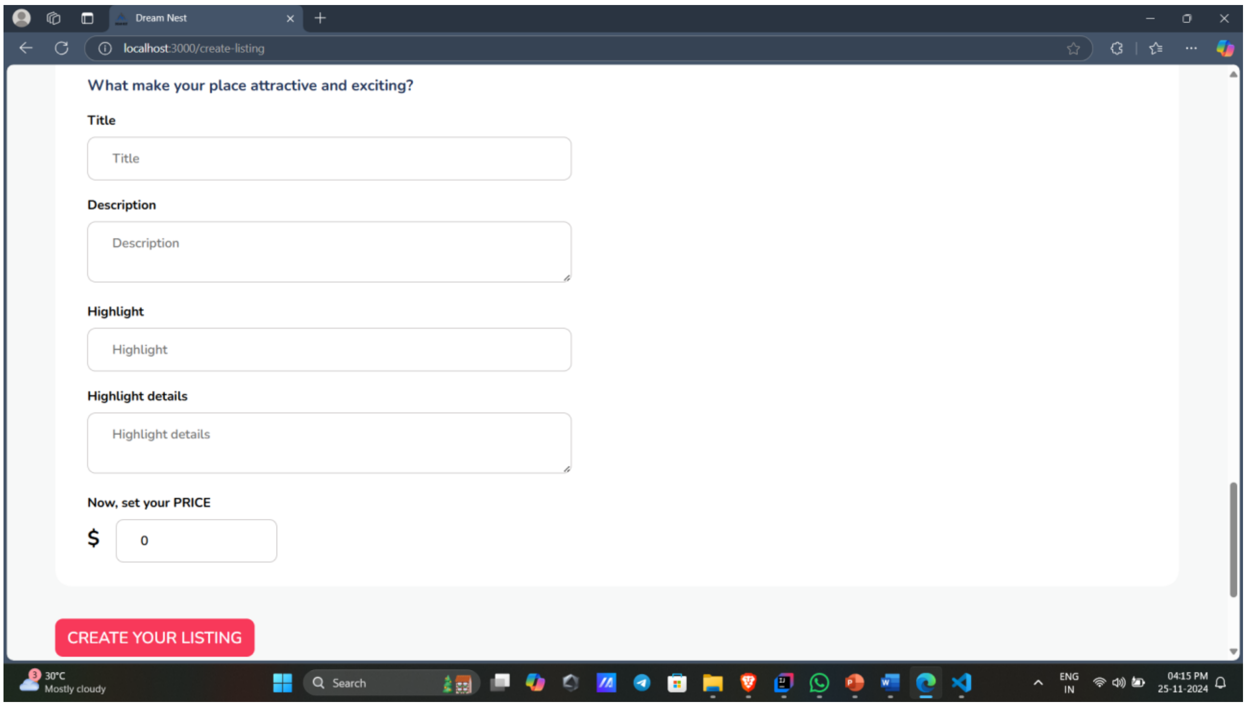
Street Address
Street Address

Apartment, Suite, etc. (if applicable)
Apt, Suite, etc. (if applicable)

City
City

Province
Province

Country
Country


What make your place attractive and exciting?

Title
Title

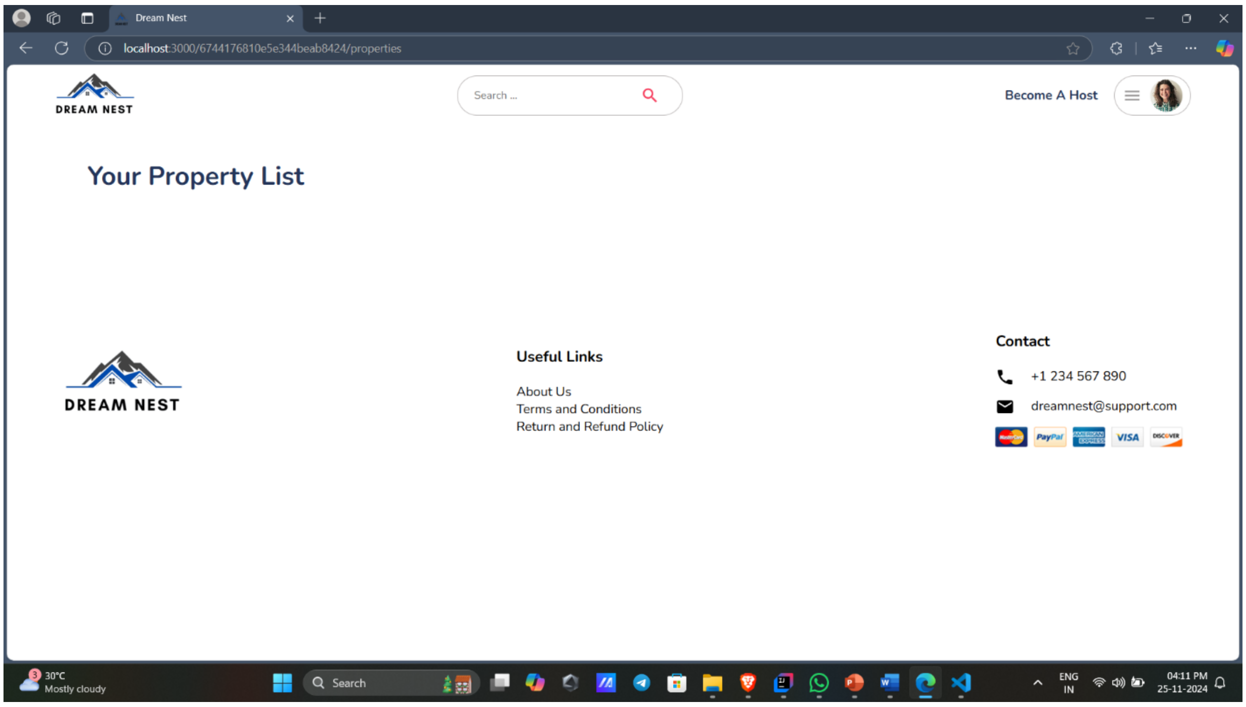
Description
Description

Highlight
Highlight

Highlight details
Highlight details

Now, set your PRICE
$ 0

CREATE YOUR LISTING



Dream Nest

localhost:3000/6744176810e5e344beab8424/properties

DREAM NEST

Search ...

Become A Host

Your Property List

DREAM NEST

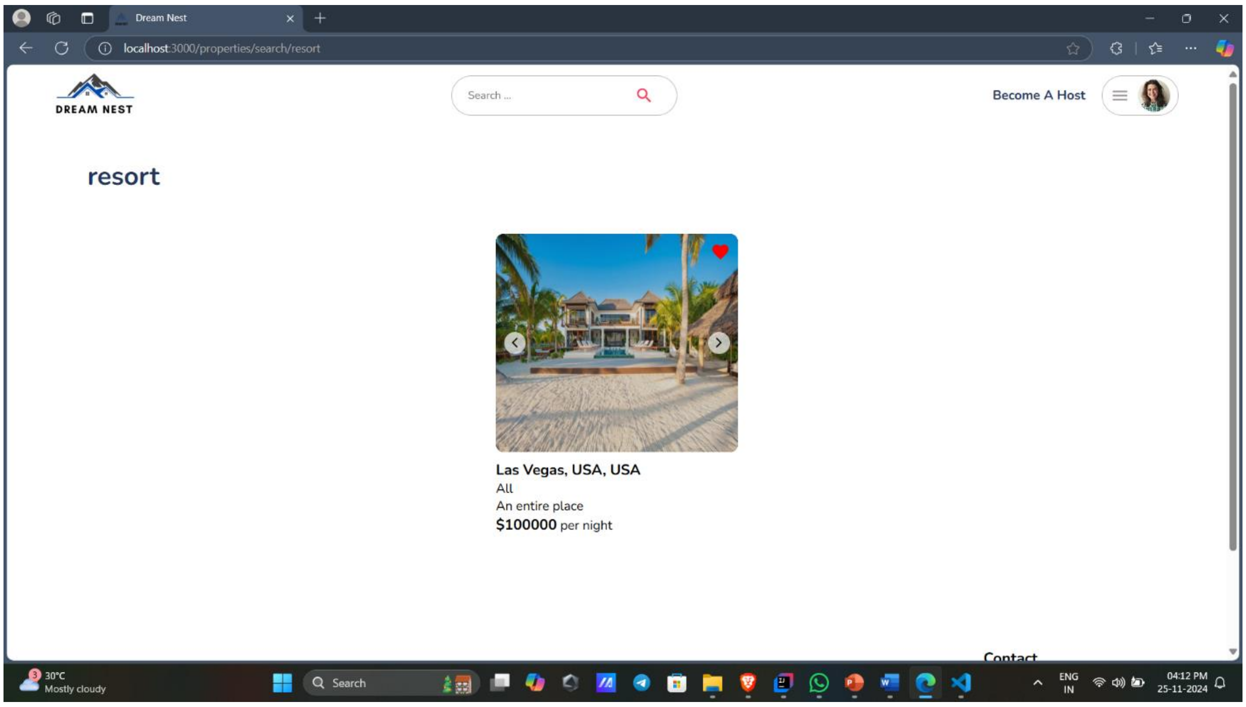
Useful Links

About Us
Terms and Conditions
Return and Refund Policy

Contact

+1 234 567 890

dreamnest@support.com



Dream Nest

localhost:3000/properties/search/resort

DREAM NEST

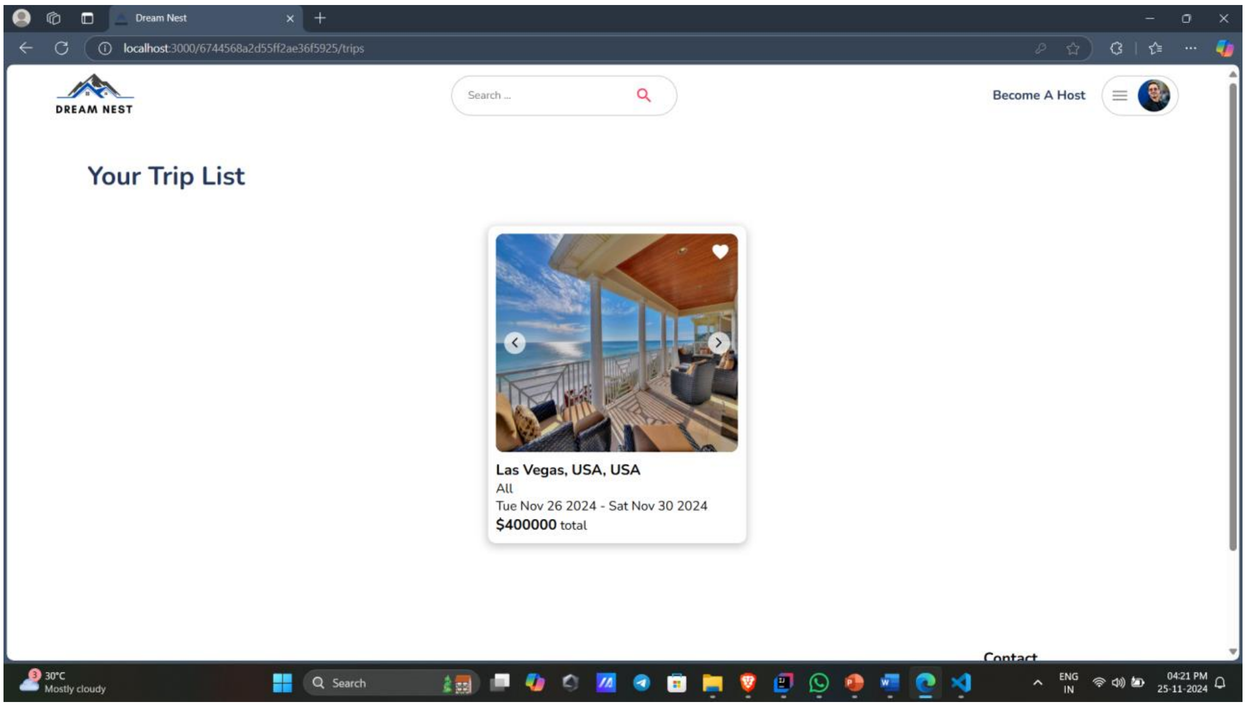
Search ...

Become A Host

resort

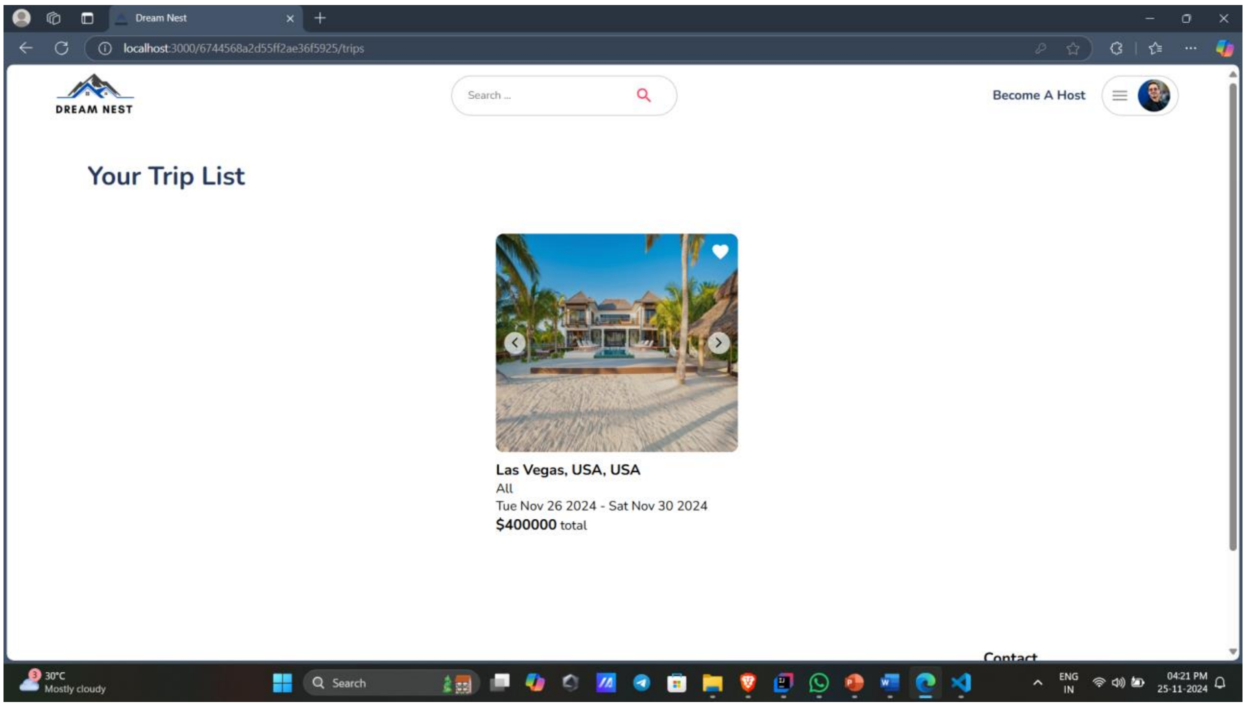
Las Vegas, USA, USA
All
An entire place
$100000 per night

Contact



Your Trip List

Las Vegas, USA, USA
All
Tue Nov 26 2024 - Sat Nov 30 2024
$400000 total



Your Trip List

Las Vegas, USA, USA
All
Tue Nov 26 2024 - Sat Nov 30 2024
$400000 total

Demo Link:

Link:https://drive.google.com/file/d/169eOLPre0ymwLzVyvUVSGCMA9QljaSOK/view?usp=drivesdk

12. Known Issues

Below is a list of known bugs or issues that users or developers should be aware of while working with the house rent app built using the MERN stack. These issues are categorized for clarity and include possible workarounds or planned fixes where applicable.

Backend Issues

a) Token Expiration Handling Issue:

- If a user's JWT expires, the app does not automatically log them out or provide a user-friendly error message.
- Impact: Users may experience unexpected behavior when attempting API requests after token expiration.

b) Data Validation Gaps Issue:

- Some API endpoints do not have strict validation, allowing invalid data to be saved (e.g., missing fields for property creation).
- Impact: Corrupt data in the database, leading to unexpected frontend errors.

c) File Uploads Issue:

- Image uploads for property listings do not handle large file sizes effectively, potentially crashing the server.
- impact: Users uploading high-resolution images may face unresponsiveness.

Frontend Issues

a) Inconsistent Error Messages

- Issue: Error messages displayed to users are inconsistent or unclear (e.g., "Something went wrong").
- Impact: Users may not understand the cause of errors, leading to frustration.

b) State Management Glitches

- Issue: State inconsistencies occur in some scenarios, such as navigating away from the booking page and returning.
- Impact: Users may see outdated or incomplete data.

c) Pagination Issues

- Issue: Pagination does not work correctly when filters (e.g., city or price range) are applied.
- Impact: Users cannot browse properties effectively.

13. Future Enhancements

1. User Features

a) Advanced Search and Filtering

- **Description:** Add advanced search capabilities allowing users to filter properties by more granular criteria (e.g., amenities, proximity to public transport, pet-friendliness, etc.).
- **Implementation Plan:**
  - Extend the MongoDB schema to include additional property attributes.
  - Update API endpoints to accept and process new filter parameters.
  - Enhance the frontend UI with additional filter controls.

b) Multi-language Support

- **Description:** Provide the app in multiple languages to cater to a diverse audience.
- **Implementation Plan:**
    - Use libraries like **i18next** for language translation in React.
    - Store translations in JSON files and allow users to switch languages dynamically

2. Property Owner Features

a) Analytics Dashboard for Property Owners

- **Description:** Provide analytics to property owners showing views, bookings, and revenue for their properties.
- **Implementation Plan:**
    - Implement aggregation queries in MongoDB to compute analytics.
    - Create a dedicated dashboard with graphs and charts using **Chart.js** or **Recharts**.

b) Dynamic Pricing Suggestions

- **Description:**
  This feature will provide property owners with real-time, data-driven suggestions for optimal pricing of their properties. The recommendations will be based on factors such as current market trends, demand patterns, seasonal variations, and property-specific attributes (e.g., location, amenities). By implementing this feature, property owners can maximize occupancy rates and revenue while remaining competitive.
- **Implementation Plan:**

- **Data Collection**
    - **External APIs:** Integrate with APIs providing real estate market trends and local demand data.
    - **Historical Data:** Leverage property booking and pricing history stored in MongoDB.

- **Pricing Algorithm**
    - Develop a dynamic algorithm that combines external market data and internal historical data.
    - Use machine learning or statistical models to analyze patterns and predict optimal pricing.

- **Integration into Dashboard**
    - Design a property management dashboard section for pricing suggestions.
    - Use data visualization tools like Chart.js or Recharts to display trends and recommendations.
    - Provide actionable insights (e.g., "Increase price by 10% during peak season").

- **Testing and Iteration**
    - Test the algorithm with real-world data and refine based on owner feedback.
    - Monitor the impact of suggestions on booking rates and revenue.

- **User Experience**
    - Ensure the suggestions are intuitive, with explanations for the recommended price adjustments.
    - Allow property owners to manually adjust prices if needed, with the suggested range as guidance.

3.Social and Community Features

a) User Reviews and Ratings

- **Description:** Allow users to leave reviews and ratings for properties after staying.
- **Implementation Plan:**
  - Add a "reviews" collection in the database.
  - Display average ratings and individual reviews on property detail pages.

b) Chat Between Users and Property Owners

- **Description:** Enable real-time chat between users and property owners to discuss details.
- **Implementation Plan:**
  - Use **Socket.io** or **Firebase Realtime Database** for chat functionality.
  - Implement chat UI with features like message history and notifications.