HouseHunt –Web Application Documentation

INTRODUCTION

House Hunt is a web-based rental platform built on the **MERN stack (MongoDB, Express.js, React.js, Node.js)**. It provides a seamless experience for renters and property owners to connect, interact, and finalize rental deals. House Hunt offers intuitive property browsing, real-time inquiry systems, and a streamlined booking experience — all under the supervision of an admin panel that ensures security, verification, and compliance.

The platform aims to simplify the process of **finding, listing, and renting properties** by offering personalized dashboards for renters, owners, and admins.

Description

Welcome to **House Hunt**, a modern rental platform that redefines how renters connect with verified property owners. Our intuitive interface enables users to browse, search, and book residential rental properties effortlessly—ranging from compact studios to spacious family homes—all tailored to individual needs and preferences.

At House Hunt, we prioritize **convenience, transparency, and security** throughout the rental process. Renters can explore detailed property listings, view photos, compare rent options, and reach out directly to property owners. Once a suitable property is found, renters can submit inquiries and manage the entire booking process within the platform—ensuring a smooth, end-to-end experience.

Our dedicated admin team plays a vital role in maintaining platform integrity. By verifying property owner profiles and listings, admins ensure all content is legitimate, safe, and up to quality standards. They also monitor interactions and provide support to create a trustworthy environment for all users.

Property owners benefit from a user-friendly dashboard where they can list properties, update availability, and manage renter inquiries. Once a rental agreement is finalized, owners can track the booking status and receive feedback—all within the House Hunt platform.

Stay updated with **real-time notifications** about new listings, booking statuses, and platform updates. Whether you're a renter searching for a place to call home or an owner looking to fill vacancies, House Hunt provides a seamless solution that connects people, homes, and peace of mind.

Join House Hunt today and experience a better way to find, manage, and rent properties in a secure and dynamic digital environment.

Scenario based case-study

Sarah, a recent graduate who just moved to a new city for her first job, is looking for a clean, affordable, and conveniently located apartment. While searching online, she discovers **House Hunt**, a user-friendly platform designed to simplify the rental process.

Finding the Perfect home: Impressed by the platform's clean design and powerful filters, Sarah begins browsing through available rental listings. She applies filters based on her budget, preferred location near her office, and a requirement for a 1BHK unit. She quickly finds a well-described listing from a property owner offering a fully furnished apartment with all the amenities she's looking for.

Inquiring with Confidence: The detailed listing includes rent amount, available move-in date, photos, and the owner's contact profile. Confident this could be the right fit, Sarah uses House Hunt's built-in inquiry form to send a message to the property owner expressing her interest and sharing a few basic details about herself.

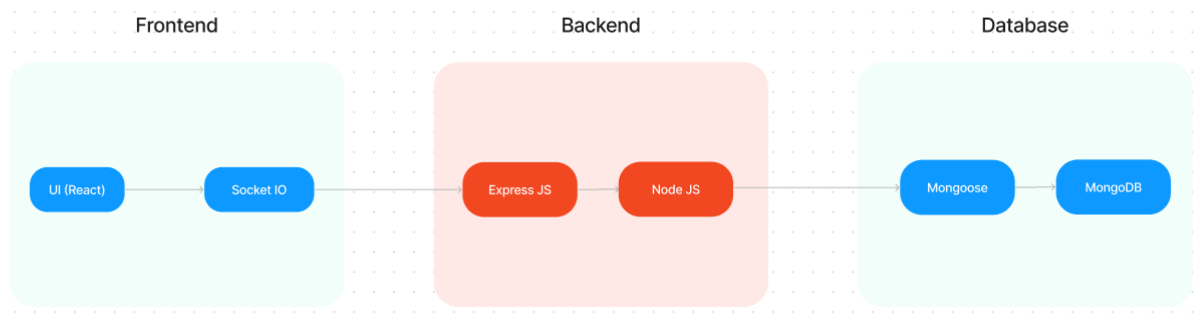
Communication & booking: The property owner, impressed with Sarah's profile and prompt communication, responds through the platform. They discuss rental terms, lease conditions, and availability via House Hunt's secure messaging system.

Finalizing the Rental: Sarah and the owner finalize the rental terms within the platform. She uploads necessary documents, reviews the lease terms, and agrees to proceed. The owner confirms her move-in date, and the booking is marked as completed.

Starting her new journey: Sarah successfully moves into her new apartment on the scheduled date. She leaves a positive review about her experience, complimenting the ease of communication and clarity of the listings on House Hunt. Satisfied with the platform, she recommends it to her co-workers and friends moving to the city.

Thanks to House Hunt, Sarah found her ideal home without brokers, confusion, or hassle—just simple, secure renting.

TECHNICAL ARCHITECTURE

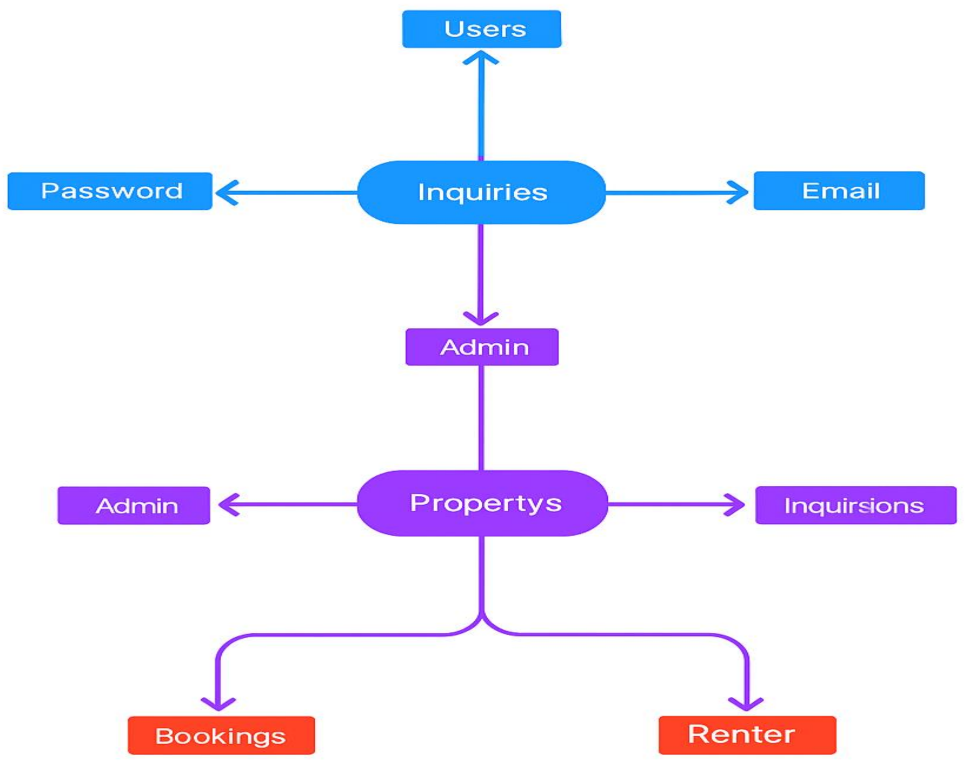


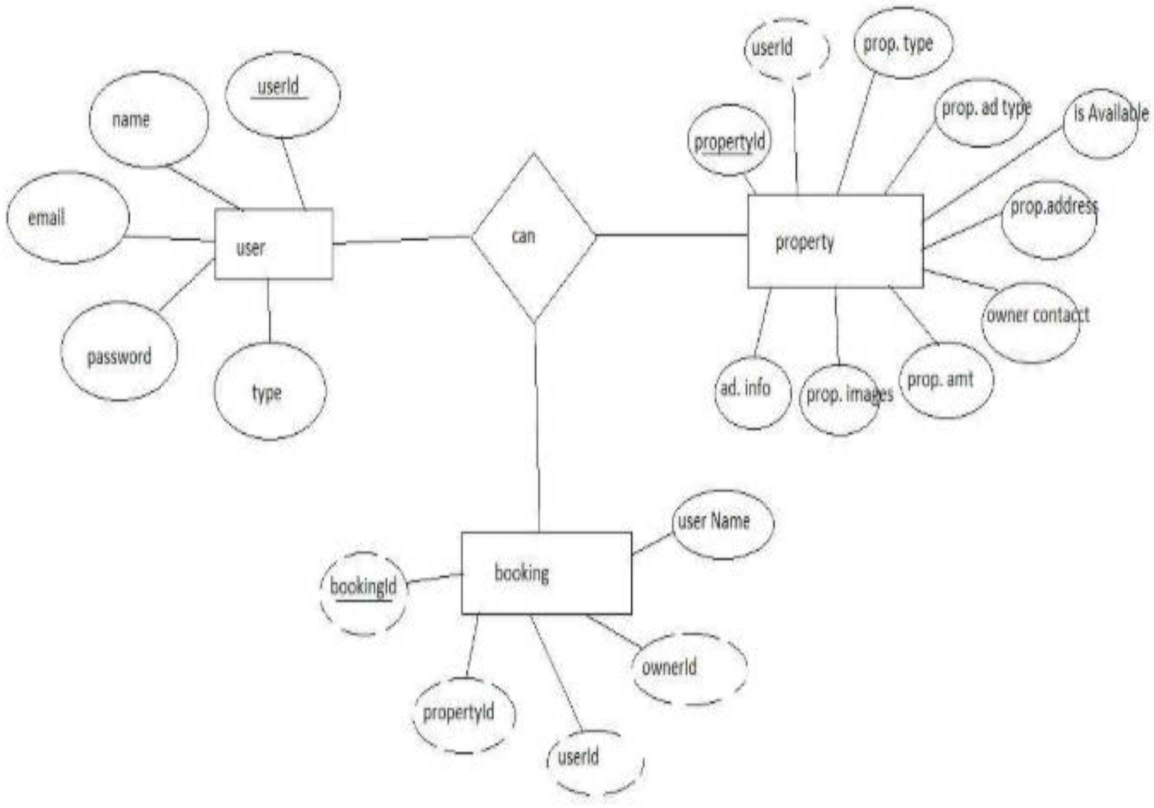
The technical architecture of House Hunt follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses the user interface and presentation layer, integrating the axios library to enable smooth communication with the backend through restful APIs. To enhance the user experience, the frontend uses Bootstrap and Material UI libraries, providing a responsive and visually appealing interface for renters, property owners, and admins.

On the backend, the application utilizes the Express.js framework to manage server-side logic and handle HTTP requests and responses efficiently. For data storage and retrieval, House Hunt relies on MongoDB, which offers a scalable and flexible solution for storing user data, property listings, booking details, and communication history. This ensures quick and reliable access to the information needed throughout the rental process.

Together, the frontend and backend components, powered by React.js, Express.js, and MongoDB, form a comprehensive technical architecture for House Hunt. This architecture enables real-time interaction, efficient data exchange, and seamless integration, delivering a smooth and secure rental experience for all users.

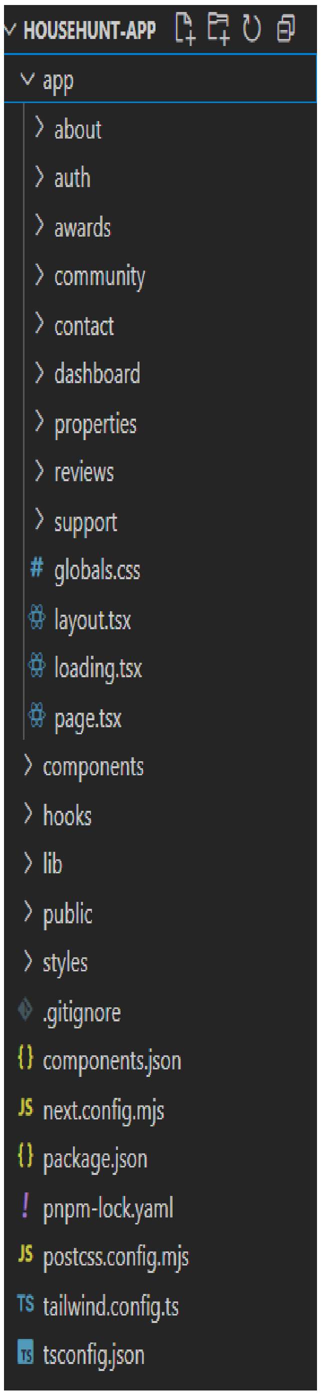
ER DIAGRAM



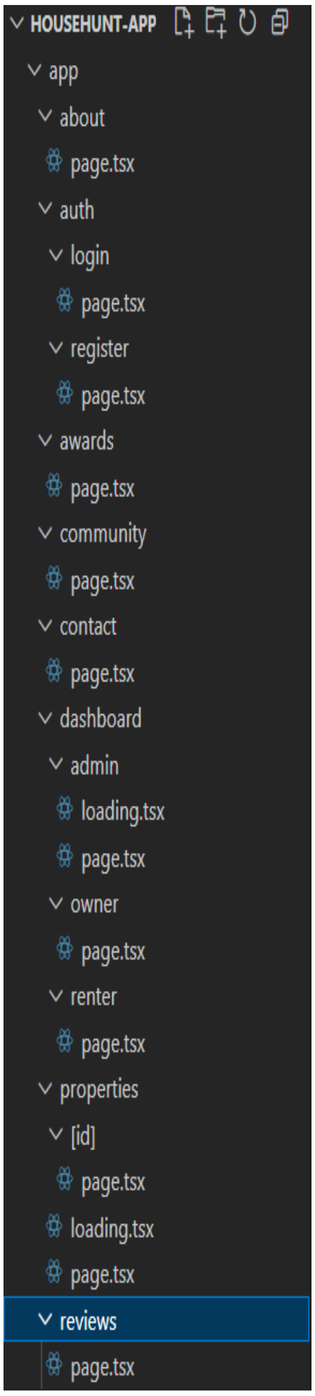


House Hunt connects renters with verified property owners through a user-friendly platform. Renters can browse property listings with detailed information and filter results to find the perfect home. Property owners can list their rentals, manage availability, and communicate directly with potential tenants through secure messaging. An admin team ensures listing authenticity, smooth interactions, and platform safety, making House Hunt a trusted solution for both renters and property owners.

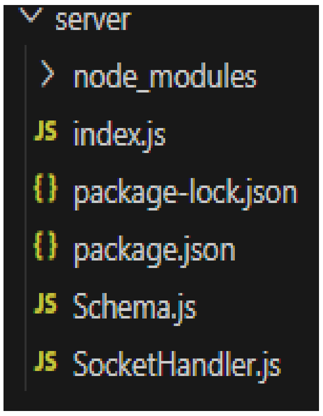
PROJECT STRUCTURE



```
v HOUSEHUNT-APP
  v app
    > about
    > auth
    > awards
    > community
    > contact
    > dashboard
    > properties
    > reviews
    > support
    # globals.css
    layout.tsx
    loading.tsx
    page.tsx
  > components
  > hooks
  > lib
  > public
  > styles
  .gitignore
  {} components.json
  JS next.config.mjs
  {} package.json
  ! pnpm-lock.yaml
  JS postcss.config.mjs
  TS tailwind.config.ts
  tsconfig.json
```



```
v HOUSEHUNT-APP
  v app
    v about
      page.tsx
    v auth
      v login
        page.tsx
      v register
        page.tsx
    v awards
      page.tsx
    v community
      page.tsx
    v contact
      page.tsx
    v dashboard
      v admin
        loading.tsx
        page.tsx
      v owner
        page.tsx
      v renter
        page.tsx
    v properties
      v [id]
        page.tsx
      loading.tsx
      page.tsx
    v reviews
      page.tsx
```



```
v server
  > node_modules
  JS index.js
  {} package-lock.json
  {} package.json
  JS Schema.js
  JS SocketHandler.js
```

The **House Hunt project** leverages React.js for the user interface. The client-side code likely consists of reusable components for **property listings, search filters, property details, and user profiles**, assembled into pages like a **search results page, a detailed property page, or a user dashboard**. Shared data like **search criteria, user authentication status, or a user's saved listings** might be managed with React Context. On the server side, Node.js handles API requests for **fetching and filtering property listings, managing user accounts, and handling property inquiries**. Mongoose models ensure structured interaction with the Mongo DB database. This breakdown provides a foundational understanding of the House Hunt project's architecture.

PRE-REQUISTIC:

Here are the key prerequisites for developing a full-stack application using Express Js, MongoDB, React.js:

✓Node.js and npm:
Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.
Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.
Download: https://nodejs.org/en/download/
Installation instructions: https://nodejs.org/en/download/package-manager/

✓Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.
Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.
Installation: Open your command prompt or terminal and run the following command:

```
npm install express
```

✓MongoDB:
MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.
Download: https://www.mongodb.com/try/download/community
Installation instructions: https://docs.mongodb.com/manual/installation/

✓React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.
Install React.js, a JavaScript library for building user interfaces.
Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

✓**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

✓**Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Express Js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations

✓**Front-end Framework**: Utilize React Js to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard. For making better UI we have also used some libraries like material UI and bootstrap.

✓**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
Git: Download and installation instructions can be found at: https://git-scm.com/downloads

✓**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from https://code.visualstudio.com/download

Install Dependencies:

• Navigate into the cloned repository directory:

```
cd freelancer-app-MERN
```

• Install the required dependencies by running the following commands:

```
cd client
npm install
../cd server
npm install
```

Start the Development Server:

• To start the development server, execute the following command:

```
npm start
```

You have successfully installed and set up the SB Works application on your local machine. You can now proceed with further customization, development, and testing as needed.

Application flow:

House Hunt Responsibilities:

- **Listing Submission:** Agents/Owners are responsible for submitting complete and high-quality property listings, including accurate details, clear photos, and detailed descriptions, through the platform.

- **Compliance:** Ensure that all submitted listings adhere to platform guidelines, local real estate regulations, and accurately represent the property's status and features.

- **Effective Communication:** Actively engage in communication with prospective buyers or renters, promptly responding to inquiries, asking clarifying questions, and providing timely updates on property availability.

- **Time Management:** Manage time effectively to promptly schedule property viewings, respond to all messages, and update the status of listings (e.g., 'pending', 'under contract', or 'sold') in a timely manner.

- **Professionalism:** Conduct oneself professionally by maintaining a respectful and cooperative attitude with users (potential buyers/renters) and fellow agents on the platform.

- **Quality Assurance:** Deliver accurate and well-executed property information to maintain user trust and satisfaction, ensuring all details are free from errors and reflect the current state of the property.

Client Responsibilities:

- **Clear Search Criteria:** Provide a detailed and comprehensive project description, including desired location, budget, property type, and any specific requirements, to help agents find suitable listings.

- **Timely Communication:** Respond promptly to agent inquiries, providing necessary information and feedback to facilitate the property search and viewing process.

- **Financial Integrity:** Fulfill any agreed-upon platform service fees or financial commitments (e.g., for viewing appointments or applications) promptly and fairly.

- **Feedback and Evaluation:** Provide constructive feedback and evaluate the agent's performance or a property's listing accuracy, helping agents improve and providing valuable insights for the community.

Admin Responsibilities:

- **Data Oversight:** As an admin, one of your key responsibilities is to monitor and ensure the integrity and security of all data on the platform, including property listings, user profiles, and inquiry records.

- **Policy Enforcement**: Admins play a crucial role in enforcing platform policies and guidelines, such as listing accuracy standards, user conduct rules, and adherence to legal standards like Fair Housing regulations.

- **Conflict Resolution**: In the event of disputes or issues within the community, such as disagreements between a user and an agent or a reported fraudulent listing, it is the admin's responsibility to address them promptly and impartially.

- **User Support and Communication**: Admins should provide support and guidance to all users on the platform, assisting with technical issues and clarifying policies for both property seekers and agents.

Project Flow:

Demo video:

https://drive.google.com/file/d/1aqYbbHCmg_4PPi9Fb5xPMehhX7sQMNRy/view

Milestone 1: Project setup and configuration.

✓ **Folder setup:**

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

- Client folders.
- Server folders

✓ **Installation of required tools:**

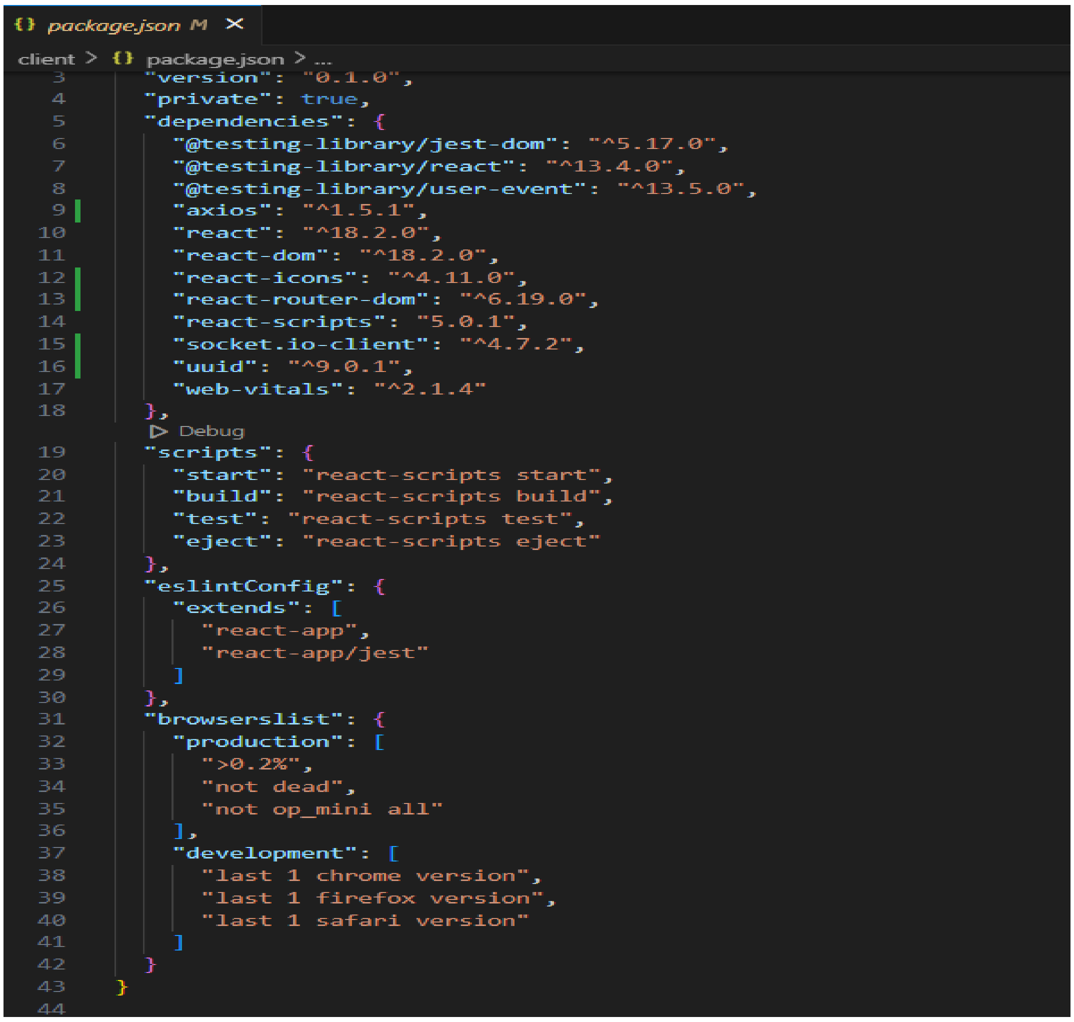
1. Open the frontend folder to install necessary tools

   For frontend, we use:
   - React
   - Bootstrap
   - Material UI
   - Axios
   - react-bootstrap

2. Open the backend folder to install necessary tools

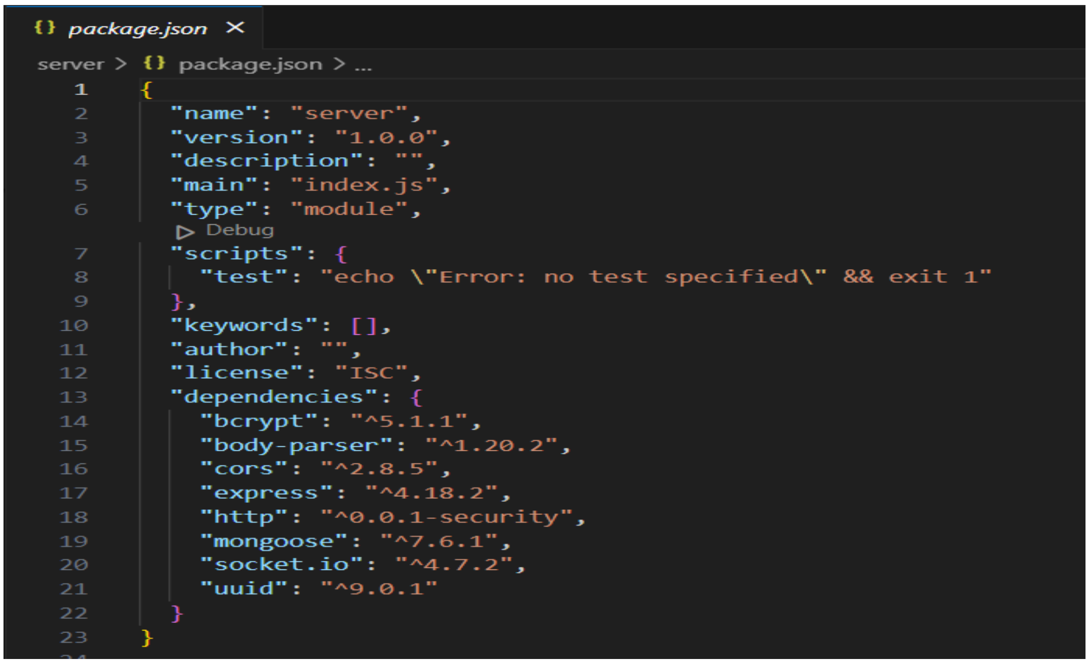
   For backend, we use:
   - Express Js
   - Node JS
   - MongoDB
   - Mongoose
   - Cors
   - Bcrypt

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.



```
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.5.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.11.0",
    "react-router-dom": "^6.19.0",
    "react-scripts": "5.0.1",
    "socket.io-client": "^4.7.2",
    "uuid": "^9.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.



```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.1",
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "http": "^0.0.1-security",
    "mongoose": "^7.6.1",
    "socket.io": "^4.7.2",
    "uuid": "^9.0.1"
  }
}
```

Milestone 2: Backend Development

1. Project Setup:

- Create a project directory and initialize it using npm init.
- Install required dependencies like Express.js, Mongoose, body-parser, and cors.

2. Database Configuration:

- Set up a MongoDB database (locally or using a cloud service like MongoDB Atlas).
- Create collections for:
- Users (storing user information, account type)
- Projects (project details, budget, skills required)
- Applications (freelancer proposals, rate, portfolio link)
- Chat (communication history for each project)
- Freelancer (extended user details with skills, experience, ratings)

3. Express.js Server:

- Create an Express.js server to handle HTTP requests and API endpoints.
- Configure body-parser to parse request bodies and cors for cross-origin requests.

4. API Routes:

- Define separate route files for user management, project listing, application handling, chat functionality, and freelancer profiles.
- Implement route handlers using Express.js to interact with the database:
- User routes: registration, login, profile management.
- Project routes: project creation, listing, details retrieval.
- Application routes: submit proposals, view applications.
- Chat routes: send and receive messages within projects.
- Freelancer routes: view and update profiles, showcase skills.

5. Data Models:

- Define Mongoose schemas for each data entity:
- User schema
- Project schema
- Application schema
- Chat schema
- Freelancer schema (extends User schema with skills, experience)
- Create Mongoose models to interact with the MongoDB database.
- Implement CRUD operations for each model to manage data.

6. User Authentication:

- Implement user authentication using JWT or session-based methods.
- Create routes and middleware for user registration, login, and logout.
- Use authentication middleware to protect routes requiring user authorization (e.g., applying for projects).

7. Project Management:

- Allow clients to post projects with details and budget.
- Enable freelancers to browse projects, search by skills, and submit proposals.
- Implement a system for clients to review applications and choose freelancers.

8. Secure Communication & Collaboration:

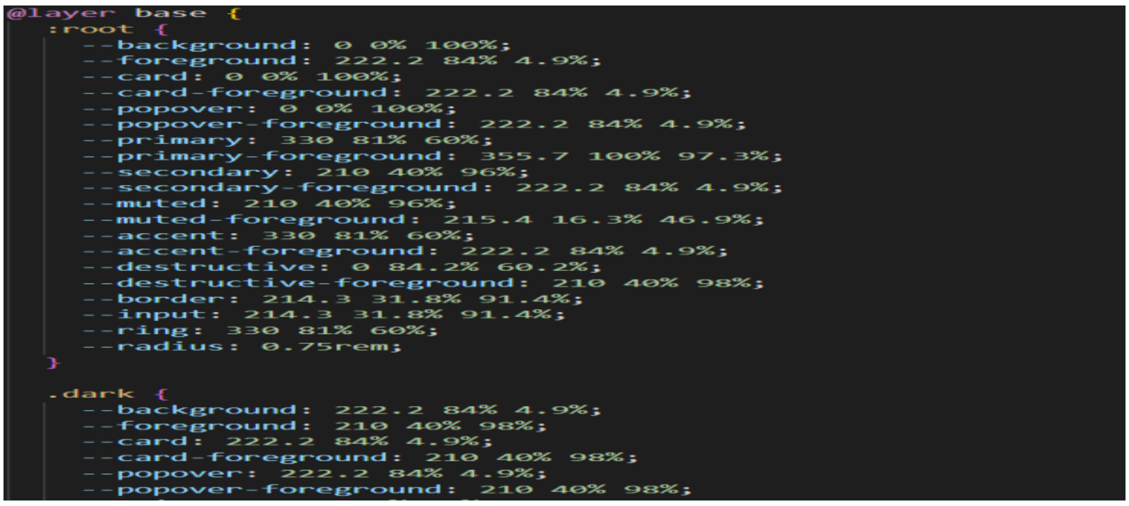
- Integrate a secure chat system within projects for communication between clients and freelancers.
- Allow file attachments and feedback exchange to facilitate collaboration.

9. Admin Panel (Optional):

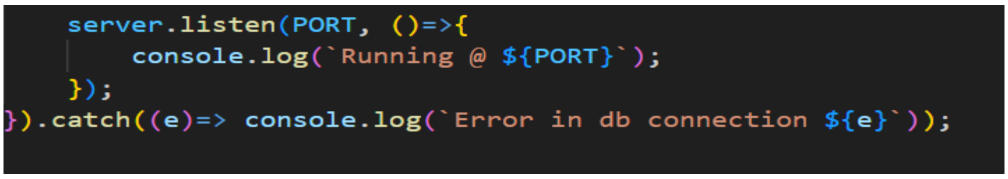
- Implement an admin panel with functionalities like:
- Managing users
- Monitoring project updates and applications
- Accessing transaction history

Milestone 3: Database development

- ○ Set up a MongoDB database either locally or using a cloud-based MongoDB servicelike MongoDB Atlas.
- ○ Create a database and define the necessary collections for users, freelancer, projects, chats, and applications.
- ○ Connect the database to the server with the code provided below.

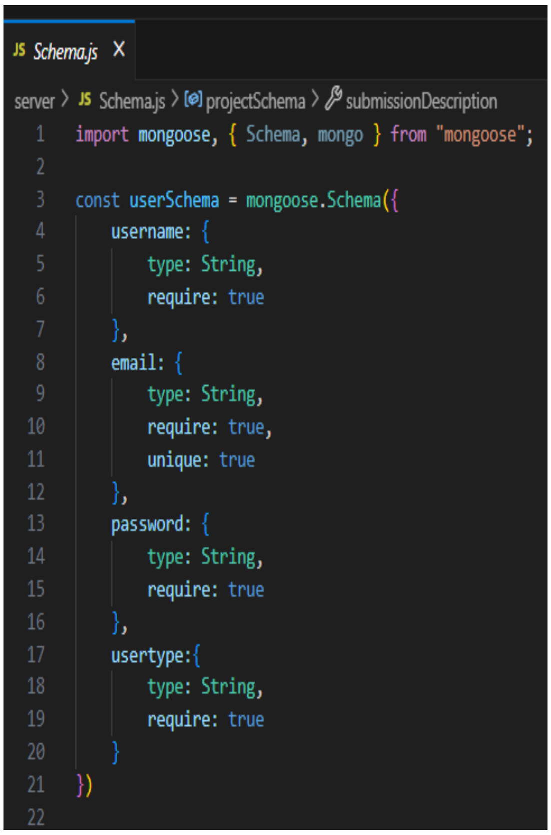


```
@layer base {
  :root {
    --background: 0 0% 100%;
    --foreground: 222.2 84% 4.9%;
    --card: 0 0% 100%;
    --card-foreground: 222.2 84% 4.9%;
    --popover: 0 0% 100%;
    --popover-foreground: 222.2 84% 4.9%;
    --primary: 330 81% 60%;
    --primary-foreground: 355.7 100% 97.3%;
    --secondary: 210 40% 96%;
    --secondary-foreground: 222.2 84% 4.9%;
    --muted: 210 40% 96%;
    --muted-foreground: 215.4 16.3% 46.9%;
    --accent: 330 81% 60%;
    --accent-foreground: 222.2 84% 4.9%;
    --destructive: 0 84.2% 60.2%;
    --destructive-foreground: 210 40% 98%;
    --border: 214.3 31.8% 91.4%;
    --input: 214.3 31.8% 91.4%;
    --ring: 330 81% 60%;
    --radius: 0.75rem;
  }

  .dark {
    --background: 222.2 84% 4.9%;
    --foreground: 210 40% 98%;
    --card: 222.2 84% 4.9%;
    --card-foreground: 210 40% 98%;
    --popover: 222.2 84% 4.9%;
    --popover-foreground: 210 40% 98%;
```



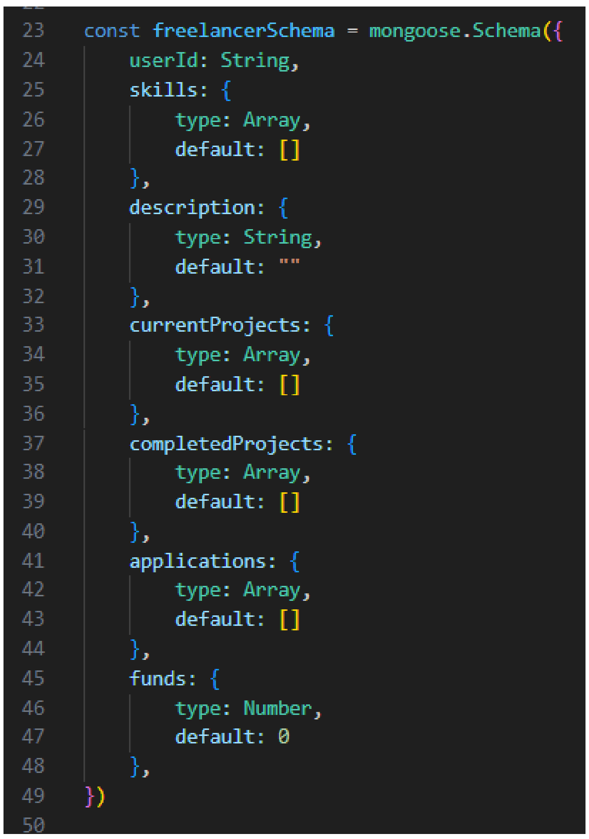
```
    server.listen(PORT, ()=>{
        console.log(`Running @ ${PORT}`);
    });
}).catch((e)=> console.log(`Error in db connection ${e}`));
```

The Schemas for the database are given below

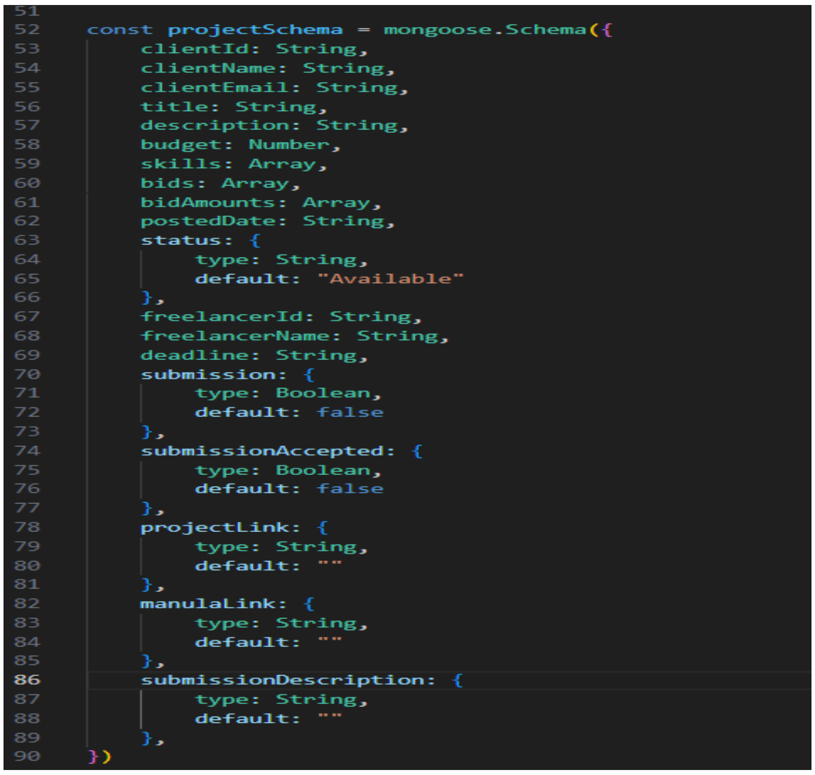


server > Schema.js > projectSchema > submissionDescription

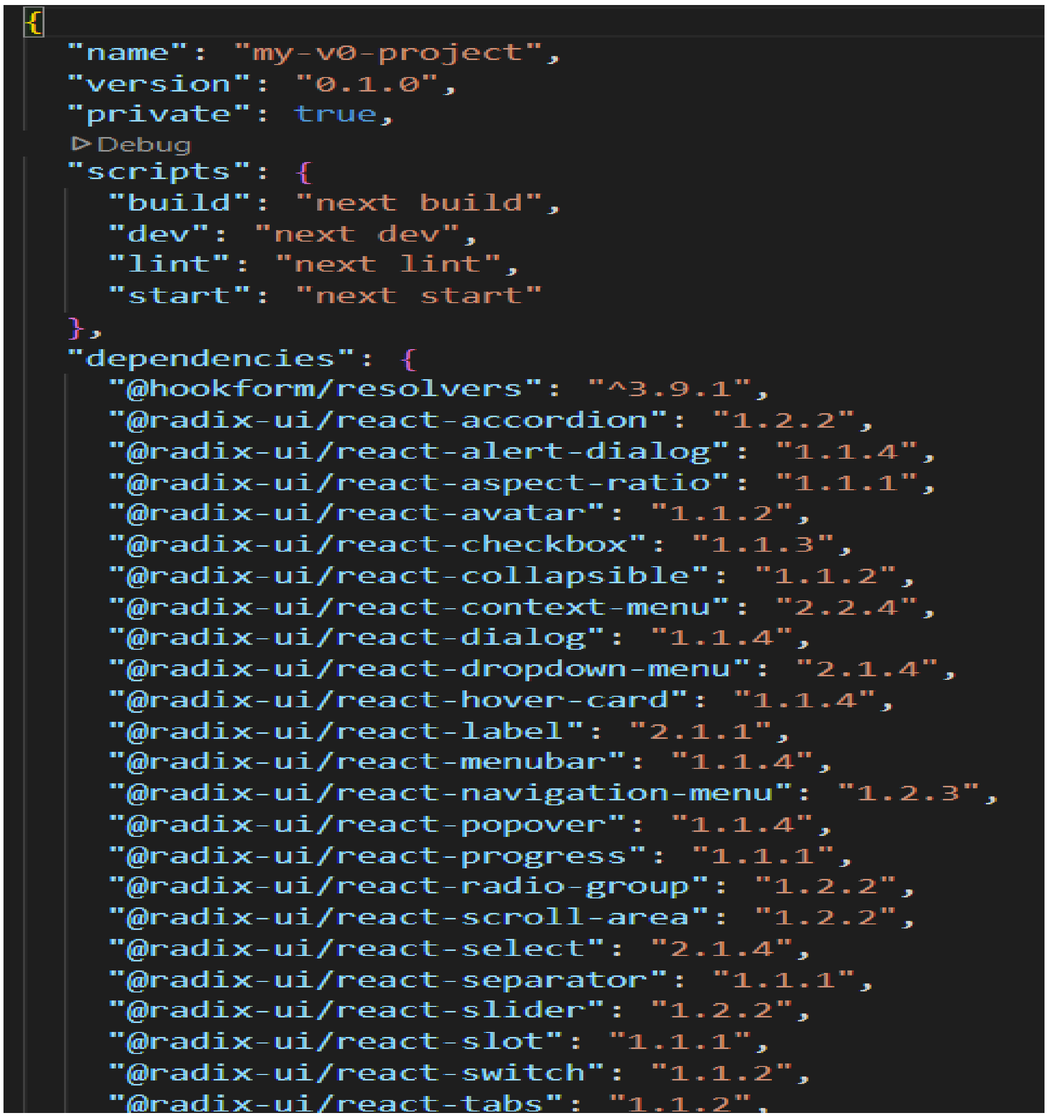
```
import mongoose, { Schema, mongo } from "mongoose";

const userSchema = mongoose.Schema({
    username: {
        type: String,
        require: true
    },
    email: {
        type: String,
        require: true,
        unique: true
    },
    password: {
        type: String,
        require: true
    },
    usertype:{
        type: String,
        require: true
    }
})

```



```
const freelancerSchema = mongoose.Schema({
    userId: String,
    skills: {
        type: Array,
        default: []
    },
    description: {
        type: String,
        default: ""
    },
    currentProjects: {
        type: Array,
        default: []
    },
    completedProjects: {
        type: Array,
        default: []
    },
    applications: {
        type: Array,
        default: []
    },
    funds: {
        type: Number,
        default: 0
    },
})

```



```

const projectSchema = mongoose.Schema({
    clientId: String,
    clientName: String,
    clientEmail: String,
    title: String,
    description: String,
    budget: Number,
    skills: Array,
    bids: Array,
    bidAmounts: Array,
    postedDate: String,
    status: {
        type: String,
        default: "Available"
    },
    freelancerId: String,
    freelancerName: String,
    deadline: String,
    submission: {
        type: Boolean,
        default: false
    },
    submissionAccepted: {
        type: Boolean,
        default: false
    },
    projectLink: {
        type: String,
        default: ""
    },
    manulaLink: {
        type: String,
        default: ""
    },
    submissionDescription: {
        type: String,
        default: ""
    },
})
```



```
{
  "name": "my-v0-project",
  "version": "0.1.0",
  "private": true,
  ▷ Debug
  "scripts": {
    "build": "next build",
    "dev": "next dev",
    "lint": "next lint",
    "start": "next start"
  },
  "dependencies": {
    "@hookform/resolvers": "^3.9.1",
    "@radix-ui/react-accordion": "1.2.2",
    "@radix-ui/react-alert-dialog": "1.1.4",
    "@radix-ui/react-aspect-ratio": "1.1.1",
    "@radix-ui/react-avatar": "1.1.2",
    "@radix-ui/react-checkbox": "1.1.3",
    "@radix-ui/react-collapsible": "1.1.2",
    "@radix-ui/react-context-menu": "2.2.4",
    "@radix-ui/react-dialog": "1.1.4",
    "@radix-ui/react-dropdown-menu": "2.1.4",
    "@radix-ui/react-hover-card": "1.1.4",
    "@radix-ui/react-label": "2.1.1",
    "@radix-ui/react-menubar": "1.1.4",
    "@radix-ui/react-navigation-menu": "1.2.3",
    "@radix-ui/react-popover": "1.1.4",
    "@radix-ui/react-progress": "1.1.1",
    "@radix-ui/react-radio-group": "1.2.2",
    "@radix-ui/react-scroll-area": "1.2.2",
    "@radix-ui/react-select": "2.1.4",
    "@radix-ui/react-separator": "1.1.1",
    "@radix-ui/react-slider": "1.2.2",
    "@radix-ui/react-slot": "1.1.1",
    "@radix-ui/react-switch": "1.1.2",
    "@radix-ui/react-tabs": "1.1.2",
```

Milestone 4: Frontend development

1. Setting the Stage:

The House Hunt project frontend thrives on React.js. To get started, we'll:

- Create the initial React application structure.
- Install essential libraries for enhanced functionality (e.g., for routing, state management, or API calls).
- Organize project files for a smooth development experience.
- This solid foundation ensures an efficient workflow as we bring the House Hunt platform interface to life.

2. Crafting the User Experience:

Next, we'll focus on the user interface (UI). This involves:

- Designing reusable UI components like buttons, forms, property listing cards, search filters, and a map view component.
- Defining the layout and styling for a visually appealing and consistent interface.
- Implementing navigation elements for intuitive movement between features, such as the search page, detailed listing view, and user dashboard.
- These steps will create a user-friendly experience for both property seekers and agents.

3. Bridging the Gap:

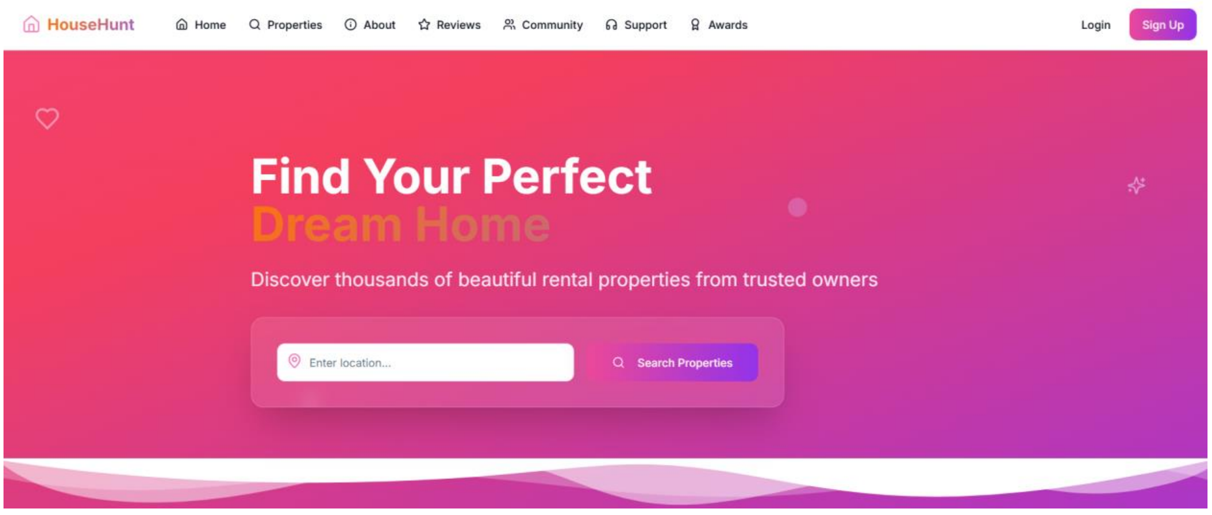
The final stage connects the visual interface with the backend data. We'll:

- Integrate the frontend with the House Hunt project's API endpoints (e.g., `/api/listings`, `/api/inquiries`).
- Implement data binding to ensure dynamic updates between user interactions (like applying filters) and the displayed information (the search results).

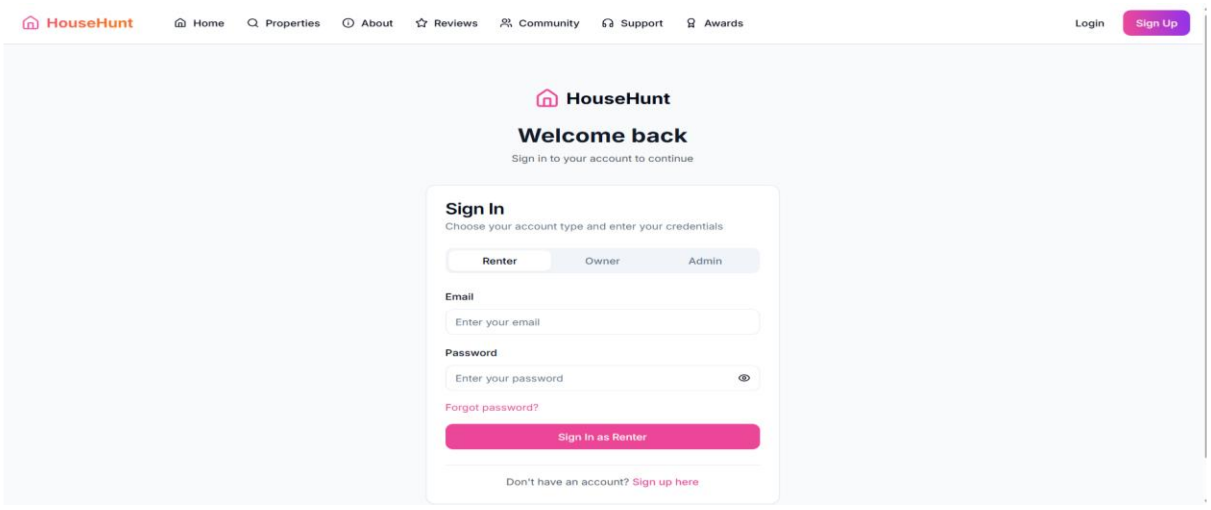
Milestone 5: Project Implementation

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the images provided below.

Landing page:



Authentication:


HouseHunt Home Properties About Reviews Community Support Awards Login Sign Up

HouseHunt

Welcome back

Sign in to your account to continue

Sign In

Choose your account type and enter your credentials

Renter Owner Admin

Email

Enter your email

Password

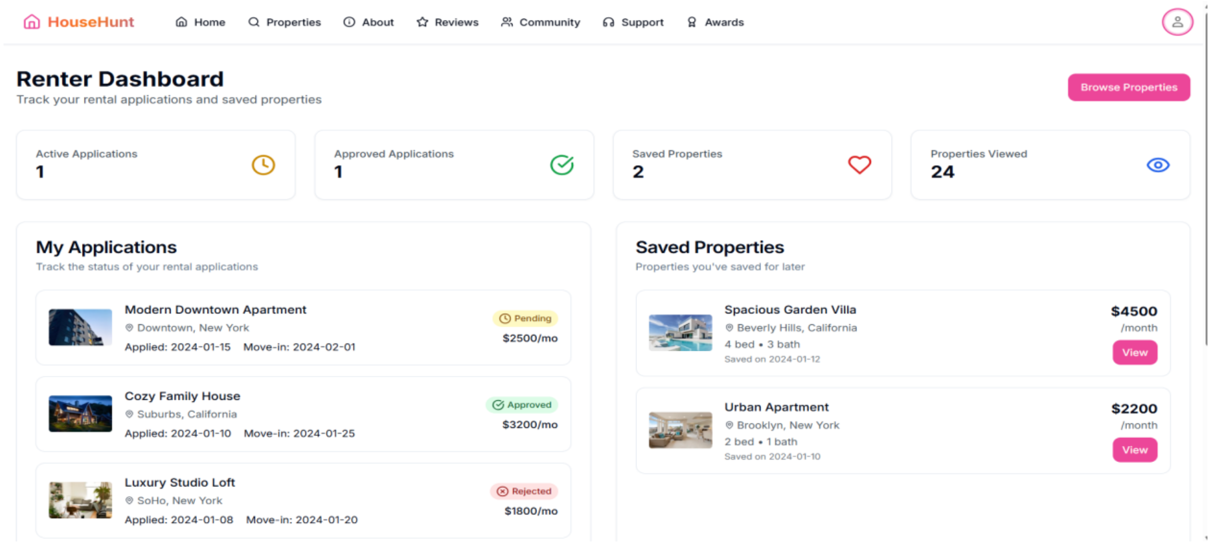
Enter your password

Forgot password?

Sign In as Renter

Don't have an account? Sign up here

Renter dashboard


HouseHunt Home Properties About Reviews Community Support Awards

Renter Dashboard

Track your rental applications and saved properties

Browse Properties

Active Applications	Approved Applications	Saved Properties	Properties Viewed
1	1	2	24

My Applications

Track the status of your rental applications

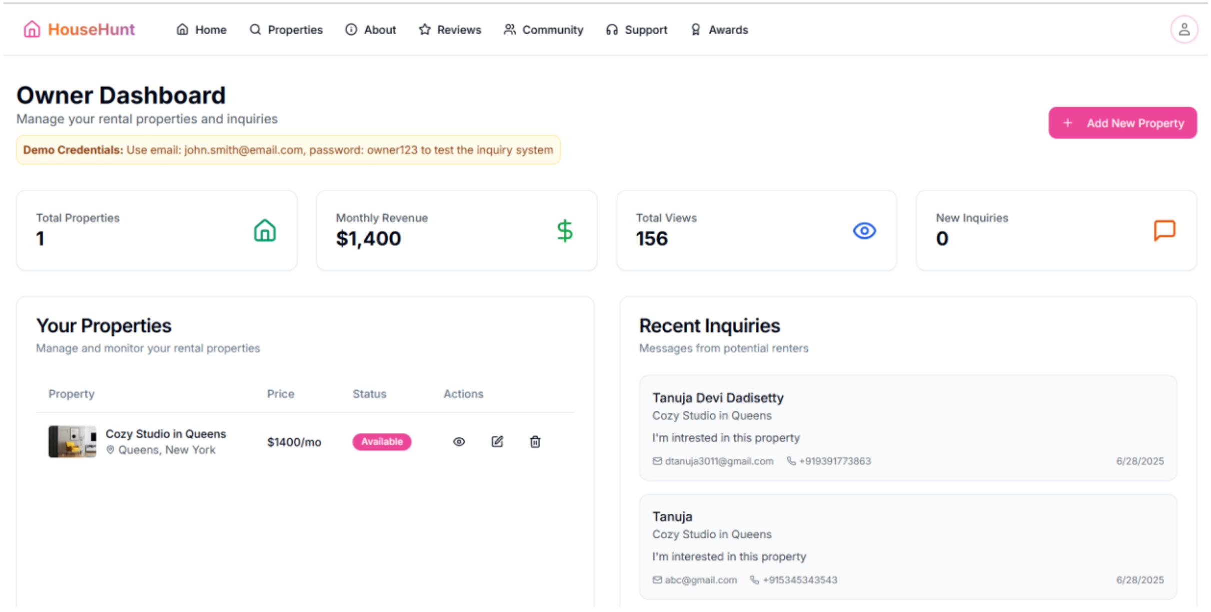
Property	Location	Applied	Move-in	Status	Price
Modern Downtown Apartment	Downtown, New York	2024-01-15	2024-02-01	Pending	$2500/mo
Cozy Family House	Suburbs, California	2024-01-10	2024-01-25	Approved	$3200/mo
Luxury Studio Loft	SoHo, New York	2024-01-08	2024-01-20	Rejected	$1800/mo

Saved Properties

Properties you've saved for later

Property	Location	Details	Saved on	Price
Spacious Garden Villa	Beverly Hills, California	4 bed • 3 bath	2024-01-12	$4500 /month
Urban Apartment	Brooklyn, New York	2 bed • 1 bath	2024-01-10	$2200 /month

Admin dashboard:


HouseHunt Home Properties About Reviews Community Support Awards

Owner Dashboard

Manage your rental properties and inquiries

Demo Credentials: Use email: john.smith@email.com, password: owner123 to test the inquiry system

+ Add New Property

Total Properties	Monthly Revenue	Total Views	New Inquiries
1	$1,400	156	0

Your Properties

Manage and monitor your rental properties

Property	Price	Status	Actions
Cozy Studio in Queens, Queens, New York	$1400/mo	Available	

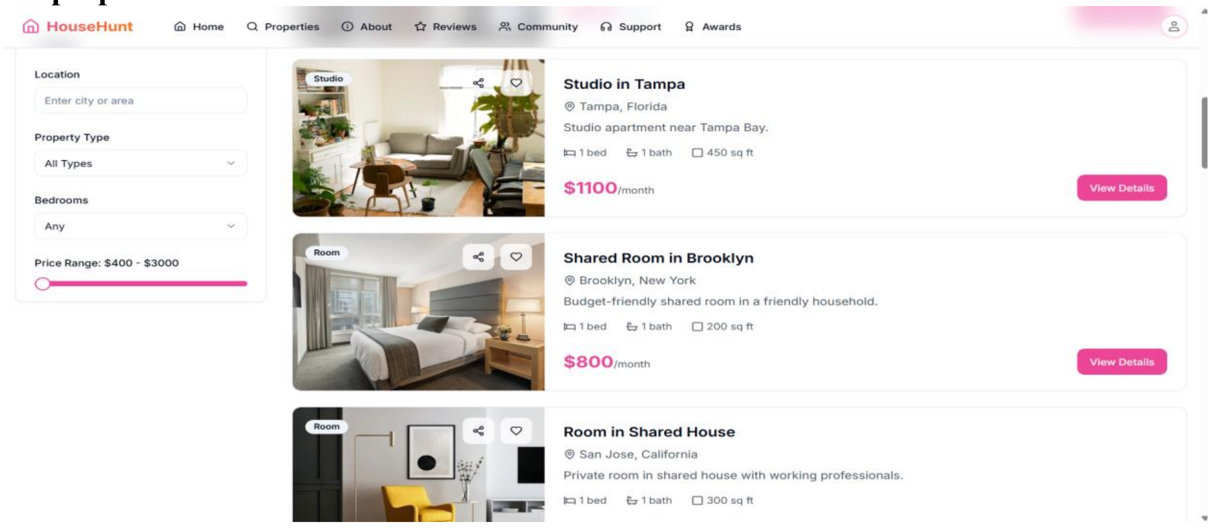
Recent Inquiries

Messages from potential renters

Tanuja Devi Dadisetty
Cozy Studio in Queens
I'm intrested in this property
dtanuja3011@gmail.com +919391773863 6/28/2025

Tanuja
Cozy Studio in Queens
I'm interested in this property
abc@gmail.com +915345343543 6/28/2025

All properties:



HouseHunt Home Properties About Reviews Community Support Awards

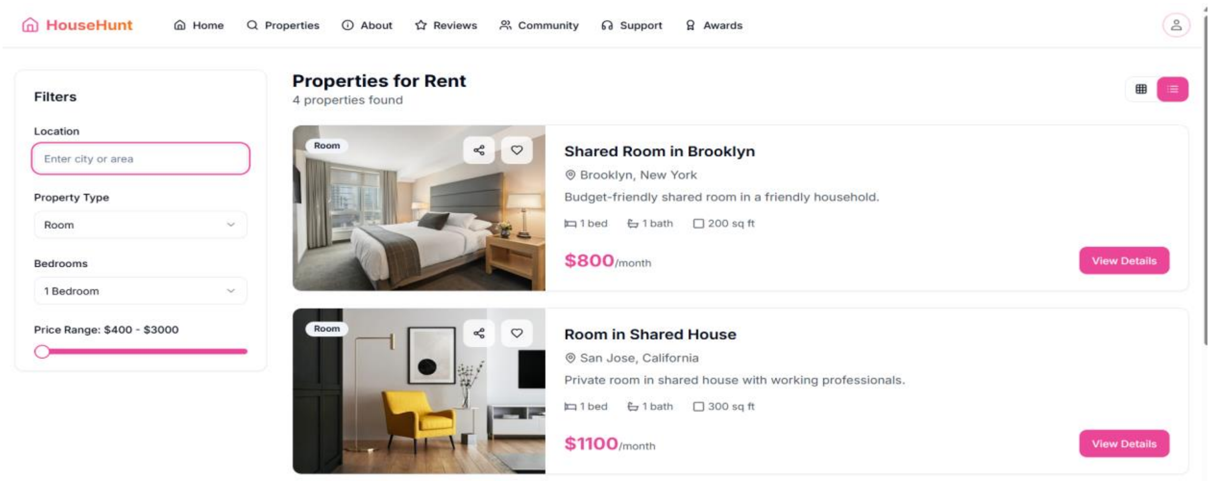
Location
Enter city or area
Property Type
All Types
Bedrooms
Any
Price Range: $400 - $3000

Studio
Studio in Tampa
Tampa, Florida
Studio apartment near Tampa Bay.
1 bed 1 bath 450 sq ft
$1100/month View Details

Room
Shared Room in Brooklyn
Brooklyn, New York
Budget-friendly shared room in a friendly household.
1 bed 1 bath 200 sq ft
$800/month View Details

Room
Room in Shared House
San Jose, California
Private room in shared house with working professionals.
1 bed 1 bath 300 sq ft

Filters:



HouseHunt Home Properties About Reviews Community Support Awards

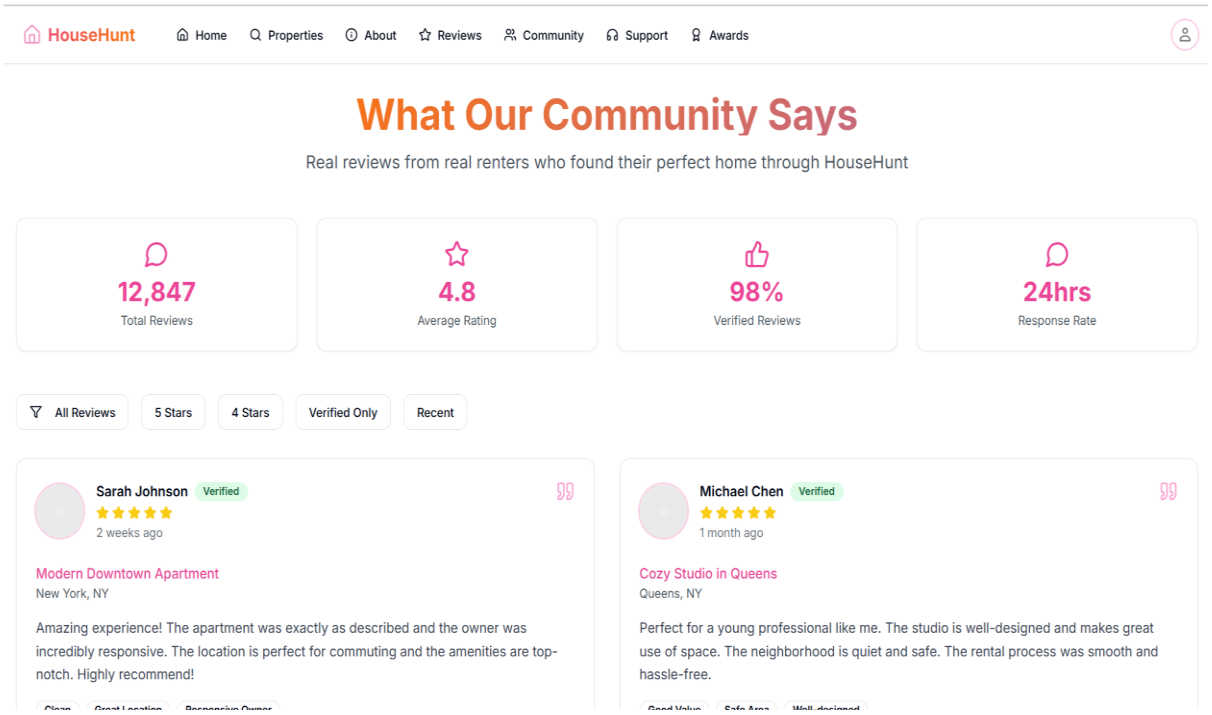
Filters
Location
Enter city or area
Property Type
Room
Bedrooms
1 Bedroom
Price Range: $400 - $3000

Properties for Rent
4 properties found

Room
Shared Room in Brooklyn
Brooklyn, New York
Budget-friendly shared room in a friendly household.
1 bed 1 bath 200 sq ft
$800/month View Details

Room
Room in Shared House
San Jose, California
Private room in shared house with working professionals.
1 bed 1 bath 300 sq ft
$1100/month View Details

Reviews:



HouseHunt | Home | Properties | About | Reviews | Community | Support | Awards

What Our Community Says

Real reviews from real renters who found their perfect home through HouseHunt

12,847
Total Reviews

4.8
Average Rating

98%
Verified Reviews

24hrs
Response Rate

All Reviews | 5 Stars | 4 Stars | Verified Only | Recent

Sarah Johnson Verified
★★★★★
2 weeks ago

Modern Downtown Apartment
New York, NY

Amazing experience! The apartment was exactly as described and the owner was incredibly responsive. The location is perfect for commuting and the amenities are top-notch. Highly recommend!

Clean | Great Location | Responsive Owner

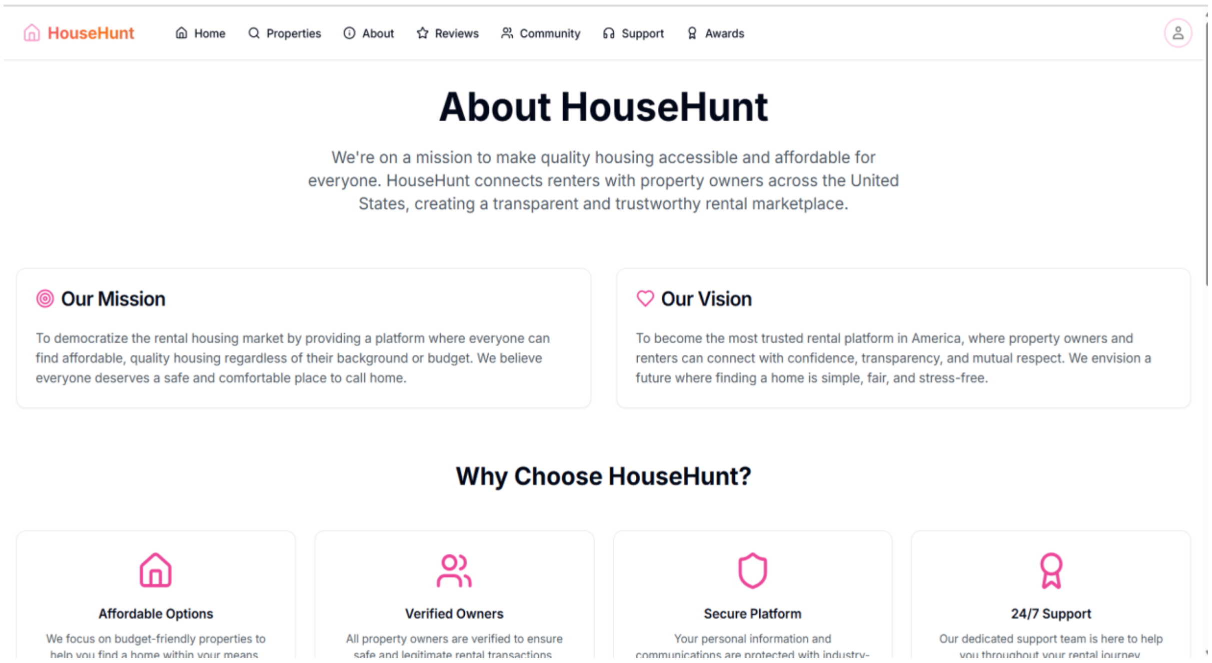
Michael Chen Verified
★★★★★
1 month ago

Cozy Studio in Queens
Queens, NY

Perfect for a young professional like me. The studio is well-designed and makes great use of space. The neighborhood is quiet and safe. The rental process was smooth and hassle-free.

Good Value | Safe Area | Well-designed

About:



HouseHunt | Home | Properties | About | Reviews | Community | Support | Awards

About HouseHunt

We're on a mission to make quality housing accessible and affordable for everyone. HouseHunt connects renters with property owners across the United States, creating a transparent and trustworthy rental marketplace.

Our Mission

To democratize the rental housing market by providing a platform where everyone can find affordable, quality housing regardless of their background or budget. We believe everyone deserves a safe and comfortable place to call home.

Our Vision

To become the most trusted rental platform in America, where property owners and renters can connect with confidence, transparency, and mutual respect. We envision a future where finding a home is simple, fair, and stress-free.

Why Choose HouseHunt?

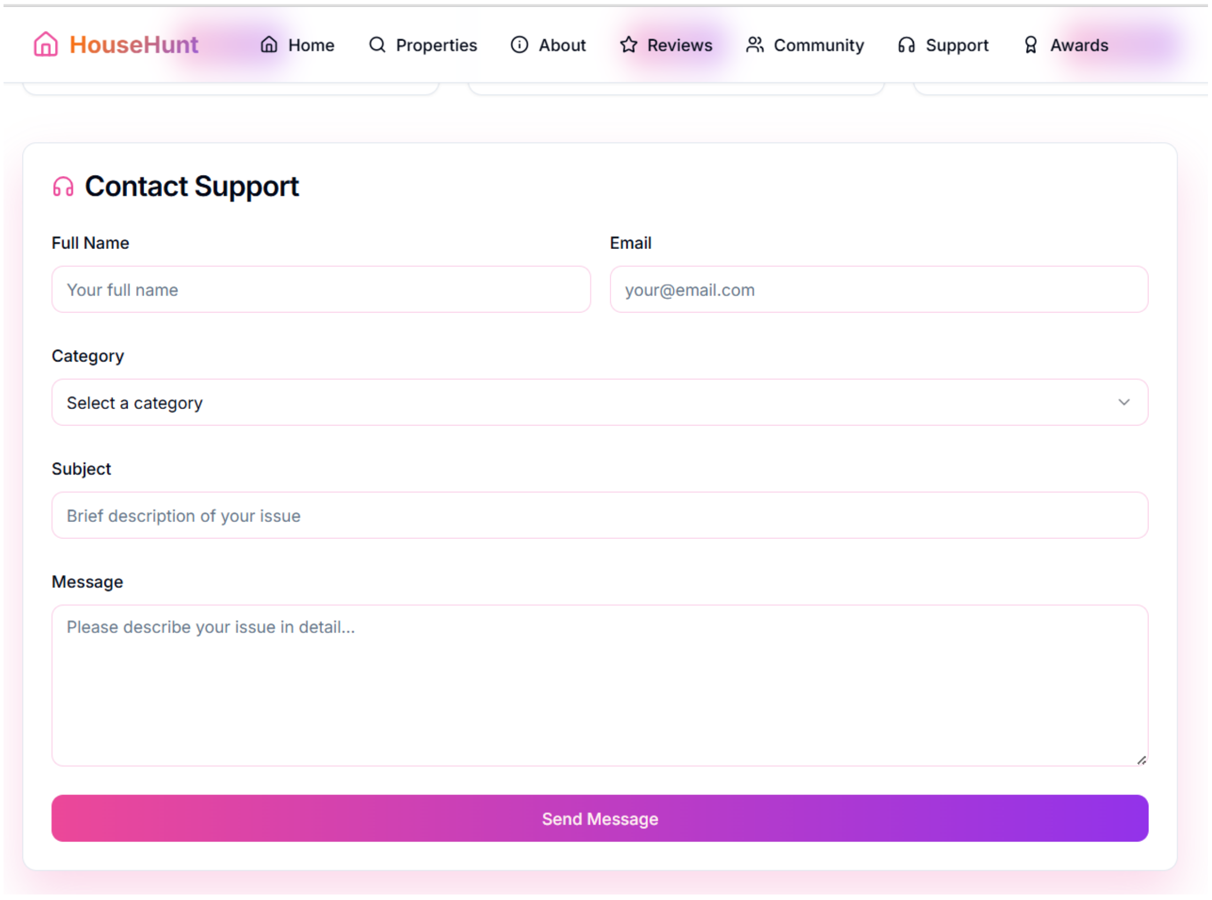
Affordable Options
We focus on budget-friendly properties to help you find a home within your means

Verified Owners
All property owners are verified to ensure safe and legitimate rental transactions

Secure Platform
Your personal information and communications are protected with industry-

24/7 Support
Our dedicated support team is here to help you throughout your rental journey

Contact:



Contact Support

Full Name: Your full name

Email: your@email.com

Category: Select a category

Subject: Brief description of your issue

Message: Please describe your issue in detail...

The demo of the app is available at:

https://drive.google.com/file/d/1aqYbbHCmg_4PPi9Fb5xPMehhX7sQMNRy/view

** THE END**