

HouseHunt: Finding Your Perfect Rental Home

Introduction

HouseHunt: Finding Your Perfect Rental Home is a smart, user-friendly web application designed to help users search for verified rental properties and manage service-related complaints. This platform enables tenants to raise issues, interact with agents, and track complaint progress. At the same time, it allows administrators and property agents to handle property-related concerns efficiently. Built using the MERN stack, HouseHunt improves the renting experience by streamlining communication and ensuring faster resolutions.

Description

HouseHunt is a comprehensive digital solution for rental property seekers. Users can search available properties, register an account, and log any issues faced during their stay. The system ensures seamless interaction between tenants and agents, with real-time updates and complaint status tracking. Admins can manage user profiles, monitor complaints, and assign agents as needed. The main objective of this platform is to increase transparency, provide a better customer experience, and manage rental-related queries with ease.

Scenario

Imagine a tenant named John who recently rented an apartment through HouseHunt. After a few days, he noticed water leakage in the bathroom. John logs into HouseHunt, navigates to the complaint section, and submits a new complaint describing the issue. The system notifies the admin, who assigns the complaint to an available agent named Sarah. Sarah contacts John through the platform's chat system, discusses the issue, and schedules a visit to inspect the problem. Once resolved, John receives a notification and provides feedback on the service. This flow shows how HouseHunt provides a practical solution for tenant-landlord communication.

Technologies Used

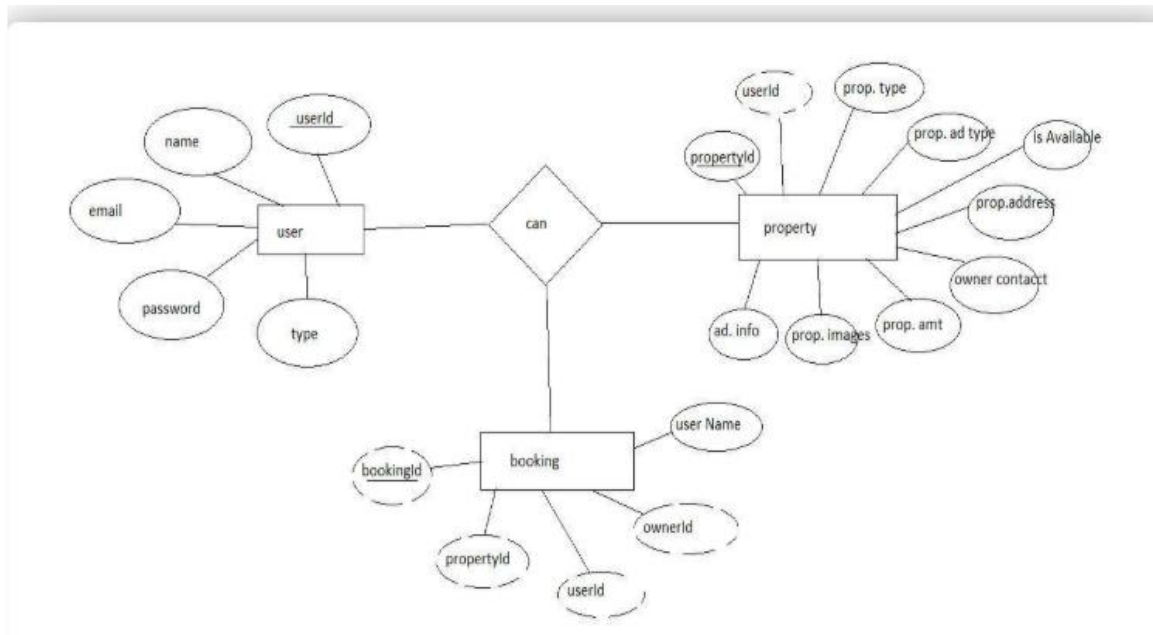
- Frontend: React.js, Bootstrap, Material UI
- Backend: Node.js, Express.js
- Database: MongoDB
- Real-time Communication: Socket.IO

- Tools: Visual Studio Code, GitHub

- Version Control: Git

- Authentication: JWT

ER DIAGRAM



This is the er diagram of the project which shows the relationship between user and agent

It shows how user which have required fields can raise a complaint by fillings required fields.

It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app. He / She can also communicate with the agent with chat window which follows the message schema which uses userid and complaintId from other schemas.

PRE-REQUISITES:

Here are the key prerequisites for developing a full-stack application using Node.js, 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 Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link:

<https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb/>

Front-end Framework: Utilize Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, 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>

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

Clone the Repository:

- Open your terminal or command prompt.
- Navigate to the directory where you want to store the e-commerce app.

Install Dependencies:

- Navigate into the cloned repository directory:

```
cd complaint-registery
```

- Install the required dependencies by running the following commands:

```
cd frontend
```

```
npm install
```

```
cd ../backend
```

```
npm install
```

Start the Development Server:

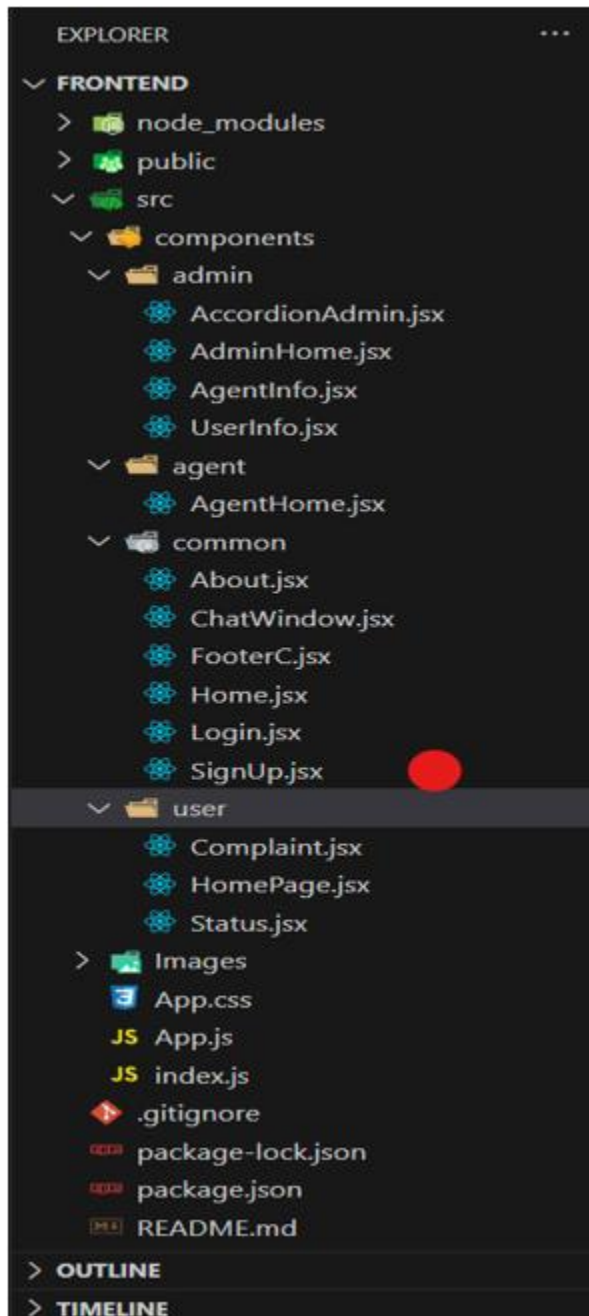
- To start the development server, execute the following command:

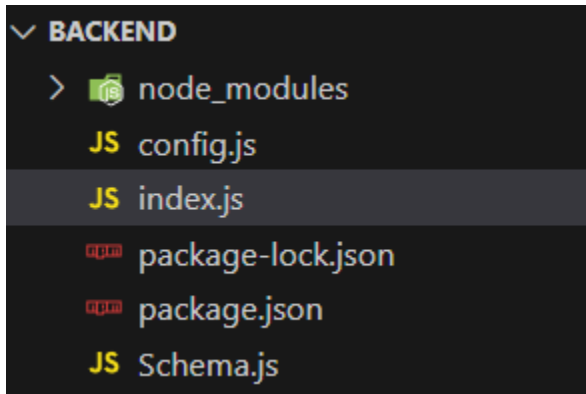
```
npm start
```

- The online complaint registration and management app will be accessible at <http://localhost:3000>

You have successfully installed and set up the online complaint registration and management app on your local machine. You can now proceed with further customization, development, and testing as needed.

Project Structure





Frontend:

- /src/components – React Components for Home, Login, Register, Dashboard
- /src/pages – Tenant, Admin, Agent views
- API integration via Axios

Backend:

- /routes – User routes, Property routes, Complaint routes
- /models – Mongoose models for User, Complaint, Property, Chat
- /controllers – Business logic for each module

Modules & Functionalities

Tenant/User:

- Register/Login
- Search rental properties
- Submit maintenance complaints
- Track complaint status
- Chat with assigned agent

Agent:

- View assigned complaints
- Communicate with tenants
- Update complaint resolution status

Admin:

- Manage tenants, agents, and complaints
- Assign complaints to available agents
- Monitor system health and user reports

Database Schemas

1. User Schema:

name, email, password, phone, userType (tenant/agent/admin)

2. Property Schema:

title, description, address, rent, postedBy (admin), availableStatus

3. Complaint Schema:

userId, propertyId, issue, status, timestamp

4. Assigned Complaint Schema:

complaintId, agentId, assignedAt, status

5. Chat Schema:

complaintId, senderId, message, time

Application Flow

Tenant Flow:

- Registers/Login
- Searches property
- Submits complaint (if required)
- Tracks complaint in dashboard
- Chats with agent

Agent Flow:

- Logs in
- Views complaints assigned
- Resolves issues, updates status
- Chats with user if needed

Admin Flow:

- Monitors all complaints
- Assigns to agents
- Manages properties and users

ER Diagram Overview

Entities:

- User → can be Admin, Agent, or Tenant
- Property → linked with Admin (posted by)
- Complaint → linked with User and Property
- Agent → handles assigned complaints
- Chat → messages linked with complaintId and userId

Project Flow

1. Setup
 - Install Node.js, MongoDB, and React
 - Create backend & frontend folders
 - Initialize Git repository

```
{
  "name": "task1",
  "version": "0.1.0",
  "proxy": "http://localhost:8000",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.2.3",
    "mdb-react-ui-kit": "^6.1.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.7.4",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.2",
    "react-scripts": "5.0.1",
    "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.

```
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "nodemon index.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "bcrypt": "^5.1.0",
14     "cors": "^2.8.5",
15     "express": "^4.18.2",
16     "express-session": "^1.17.3",
17     "mongoose": "^7.1.1",
18     "nodemon": "^2.0.22"
19   }
20 }
```

2.Backend Development

- Design models (Mongoose)
- Develop REST APIs
- Setup JWT authentication
- Integrate MongoDB

3.Database Development

- **Configure MongoDB**

1. Import mongoose.
2. Add database connection from config.js file present in config folder
3. Create a model folder to store all the DB schemas like renter, owner and booking, and properties schemas.

```

const mongoose = require('mongoose');

const connectionOfDb = () => {
  mongoose
    .connect(process.env.MONGO_DB, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    })
    .then(() => {
      console.log('Connected to MongoDB');
    })
    .catch((err) => {
      throw new Error(`Could not connect to MongoDB: ${err}`);
    });
};

module.exports = connectionOfDb;

```

4.

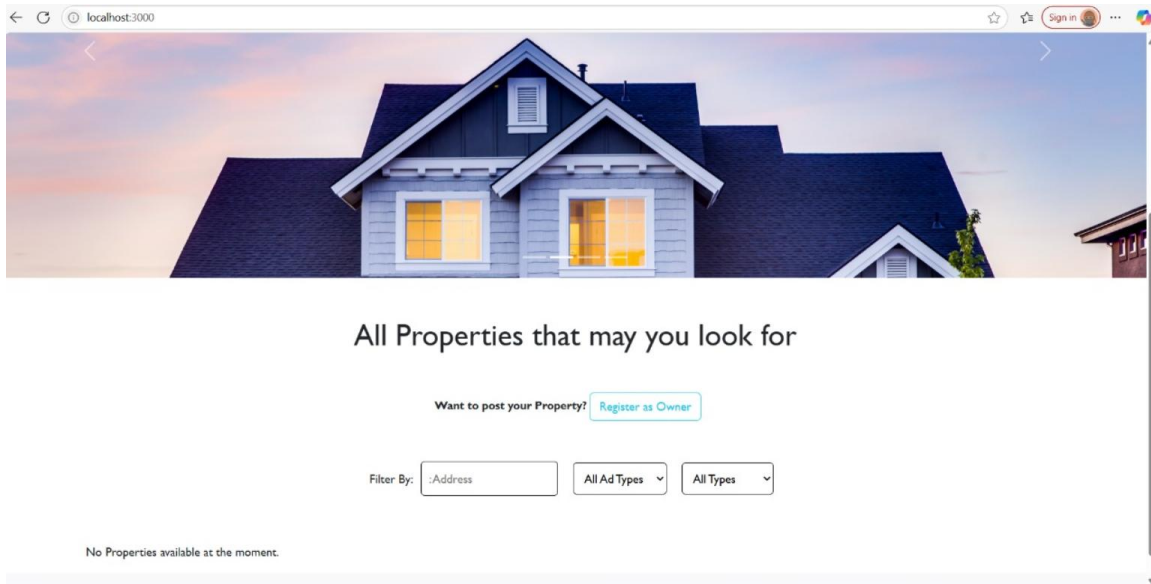
4. Frontend Development

- Build UI with React & Bootstrap
- Axios integration for API calls
- Role-based dashboards

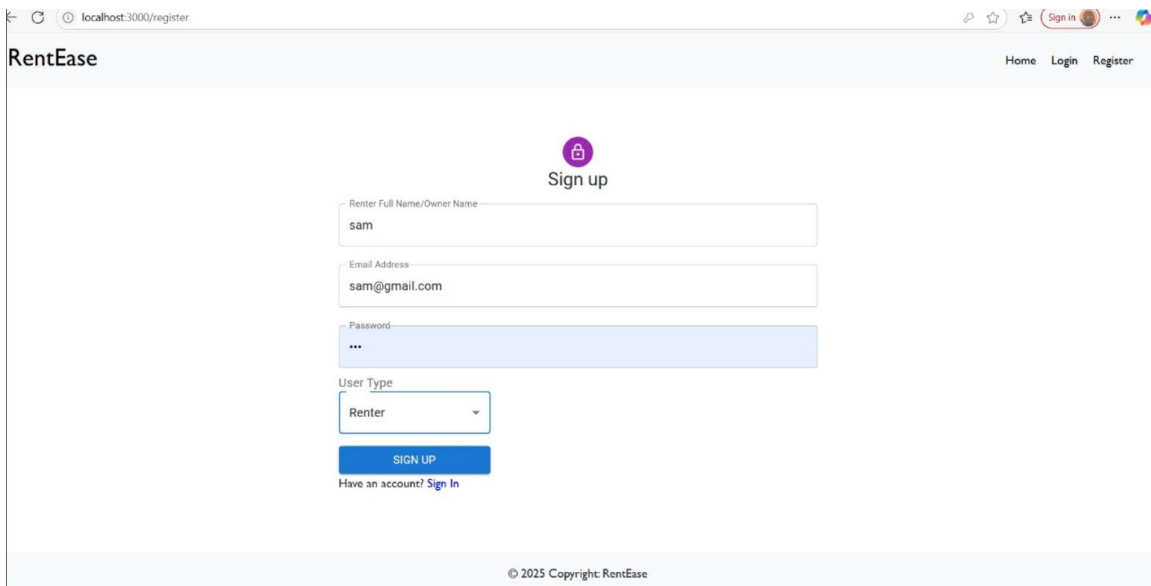
5. Project Implementation & Execution

- Test entire flow
- Run on localhost (or host on Netlify + Render)

Landing Page:



Register or Sign Up:



Login and register page:

localhost:3000/login

RentEase

Home Login Register

Sign In

Email Address
sam@gmail.com

Password

SIGN UP

forgot password? [Click here](#) Have an account? [Sign Up](#)

© 2025 Copyright: RentEase

localhost:3000/forgotpassword

RentEase

Home Login Register

Forgot Password?

Email Address
sam@mail.com

New Password

Confirm Password

CHANGE PASSWORD


Don't have an account? [Sign Up](#)

© 2025 Copyright: RentEase

16:17 25-06-2025

localhost:3000/login

RentEase



Sign In

Email Address

sam@gmail.com

Password

...

SIGN UP

forgot password? [Click here](#) Have an account? [Sign Up](#)

Admin Panel:

localhost:3000/user/home

Click to go back, hold to see history

RentEase

Hi Sam Log Out

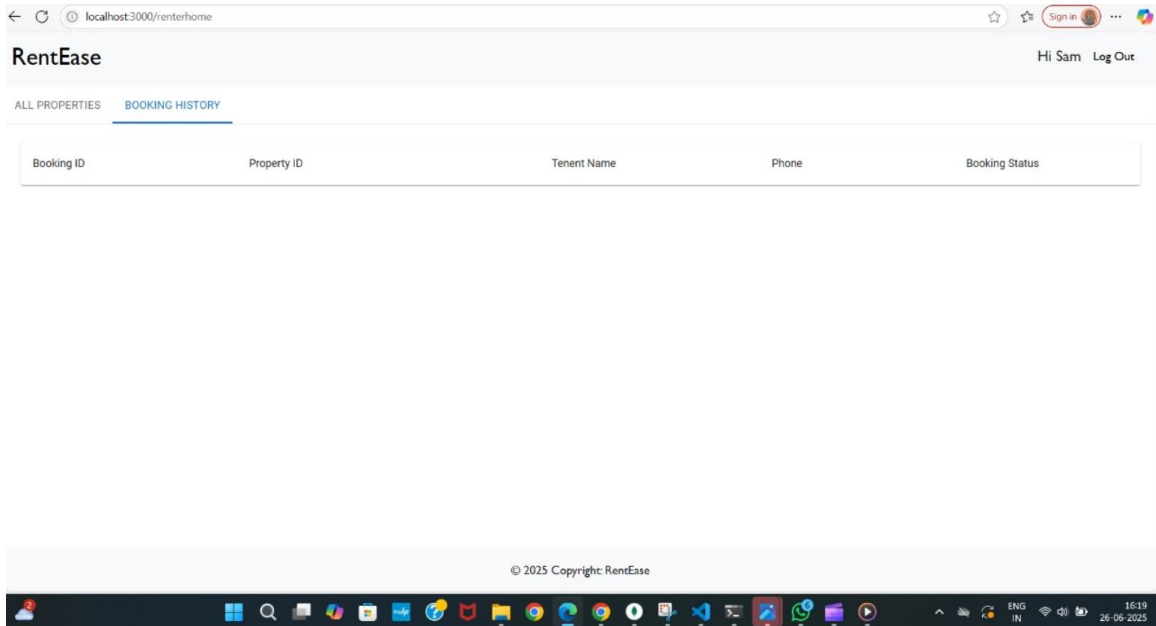
ALL PROPERTIES BOOKING HISTORY

Filter By: :Address All Ad Types All Types

No Properties available at the moment.

© 2025 Copyright: RentEase

Tenant panel:



Conclusion

HouseHunt solves real-world problems in the rental space. It provides a smooth experience for tenants to report issues and get timely resolutions. It connects tenants, agents, and admins under a single responsive web app with real-time interaction and effective complaint tracking.