

Project title :Househunt: Finding Your Perfect Rental Home

Team Members:

Team leader: Gopiseti Sujitha

Team member: Dodda Prakash

Team member : Duddukuri Vineela

Tema member : Dosapati Anjana Vishnu Priya

1.INTRODUCTION

Househunt: Finding Your Perfect Rental Home is a modern, full-stack web application that offers a centralized and user-friendly platform for individuals seeking rental properties. The system is designed to streamline the process of discovering, evaluating, and securing rental homes, while also providing landlords with powerful tools to list, manage, and communicate about their properties.

By bringing together renters and landlords in a single platform, Househunt enhances transparency, improves communication, and reduces the hassle traditionally associated with the home rental experience. The application focuses on smart filtering, location-based results, and secure interactions to ensure users can find their ideal rental property quickly and confidently.

Key Features

- 1. User Registration & Authentication:**
Users can create secure accounts to access personalized features, including saving listings, messaging landlords, and posting rental properties (for landlords).
- 2. Rental Property Listings:**
Landlords can post detailed property listings with descriptions, pricing, photos, amenities, and locations. Renters can browse and filter these listings according to their needs.
- 3. Search & Filter Functionality:**
Users can search for properties by location, price range, number of bedrooms, type of property, and more — ensuring they find homes that match their exact requirements.

4. **Favorites & Saved Listings:**
Users can save listings they're interested in for quick access later, helping them compare multiple options effectively.
5. **Communication & Messaging:**
Renters can communicate directly with landlords through a built-in messaging system, making it easy to ask questions or request visits.
6. **Listing Management Dashboard:**
Landlords have access to a dashboard where they can view and manage their listings, track inquiries, and update availability.
7. **Smart Recommendations (Optional Enhancement):**
Based on user behavior and preferences, the system can suggest properties that may be of interest to the user.
8. **Security & Privacy:**
All user data is protected through secure login mechanisms (JWT authentication), encrypted data handling, and role-based access control. Only authorized users can view or manage sensitive data.

Scenario-based Case Study:

Scenario: Renting an Apartment

User Registration: Alice, who is looking for a new apartment, downloads your house rent app and registers as a Renter. She provides her email and creates a password.

Browsing Properties: Upon logging in, Alice is greeted with a dashboard showcasing available rental properties. She can see listings with detailed descriptions, photos, and rental information. She applies filters to narrow down her search, specifying her desired location, rent range, and the number of bedrooms.

Property Inquiry: Alice finds an apartment she likes and clicks on it to get more information. She sees the property details and owner's contact information.

Interested in renting, Alice fills out a small form with her details and sends it to the owner.

Booking Confirmation: The owner receives Alice's inquiry and reviews her details. Satisfied, the owner approves Alice's booking request.

Alice receives a notification that her booking is confirmed, and the status in her dashboard changes to "pending owner confirmation."

Admin Approval (Background Process): In the background, the admin reviews new owner registrations and approves legitimate users who want to add properties to the app.

Owner Management: Bob, a property owner, signs up for an Owner account on the app and submits a request for approval.

The admin verifies Bob's credentials and approves his Owner account.

Property Management: With his Owner account approved, Bob can now add, edit, or delete properties in his account.

He updates the status and availability of his properties based on their occupancy.

Platform Governance: Meanwhile, the admin ensures that all users adhere to the platform's policies, terms of service, and privacy regulations.

The admin monitors activities to maintain a safe and trustworthy environment for all users.

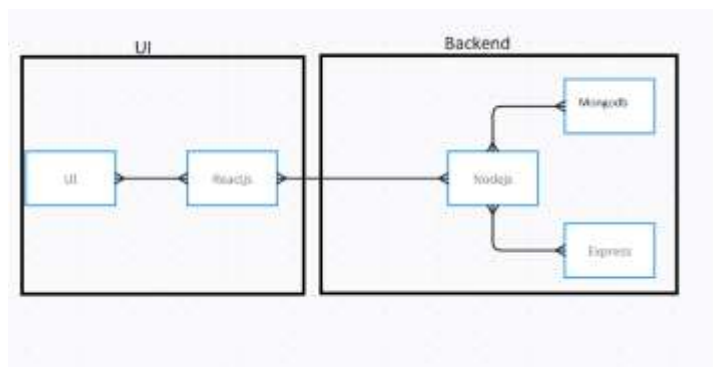
Transaction and Lease Agreement: Once Alice's booking is confirmed, she and the owner negotiate the terms of the lease agreement through the app's messaging system.

They finalize the rental contract and payment details within the app, ensuring transparency and security.

Move-in Process: Alice successfully moves into her new apartment, marking the completion of the rental process facilitated by the house rent app.

This scenario highlights the main functionalities of your MERN-based house rent app, including user registration, property browsing, inquiry and booking process, admin approval, owner management, platform governance, and the overall rental transaction.

TECHNICAL ARCHITECTURE



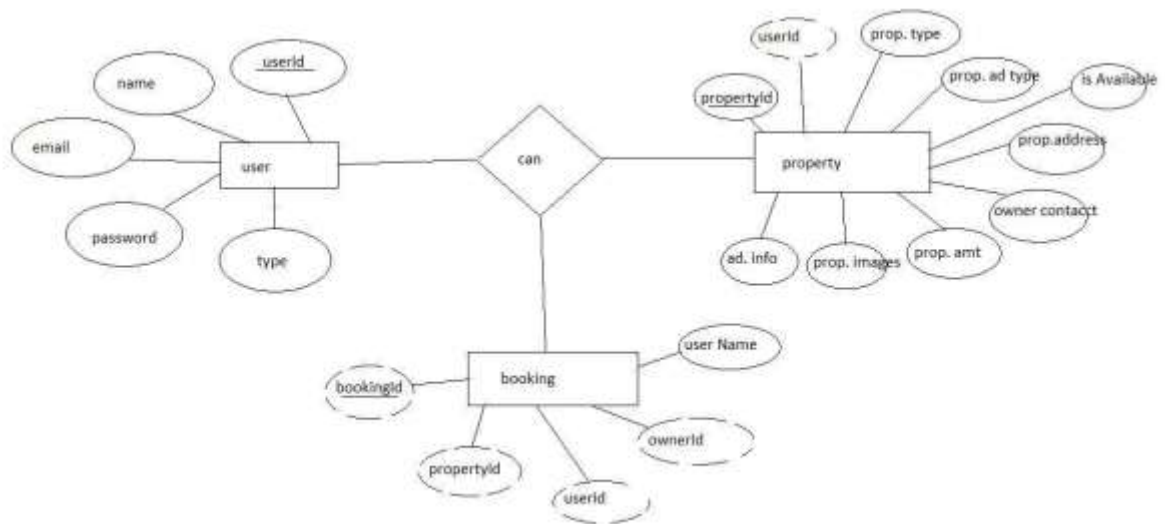
The technical architecture of our House rent app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful Apis.

The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user whether it is admin, doctor and ordinary user working on it.

On the backend side, we employ Express.js frameworks to handle the server-side logic and communication. For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, including user profiles, for booking room, and adding room, etc. It ensures reliable and quick access to the necessary information.

Together, the frontend and backend components, along with moment, Express.js, and MongoDB, form a comprehensive technical architecture for our House rent app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive booking an appointment and many more experience for all users.

ER-Diagram



Here there is 3 collections namely users, property, and booking which have their own fields in

Users:

_id: (MongoDB creates by unique default)

name

email

password

type

Property:

userID: (can be act as foreign key)

_id: (MongoDB creates by unique default)

prop.Type

prop.AdType

isAvailable

prop.Address

owner contact

prop.Amt

prop.images

add.Info

Booking

_id: (MongoDB creates by unique default)

propertId

userId

ownerId

username

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.
- Execute the following command to clone the repository:

git clone: <https://github.com/awdhesh-student/complaint-registry.git>

Install Dependencies:

- Navigate into the cloned repository directory:

`cd complaint-registry`

- Install the required dependencies by running the following commands:

`cd frontend`

`npm install`

```
cd ../backend
```

```
npm install
```

Start the Development Server:

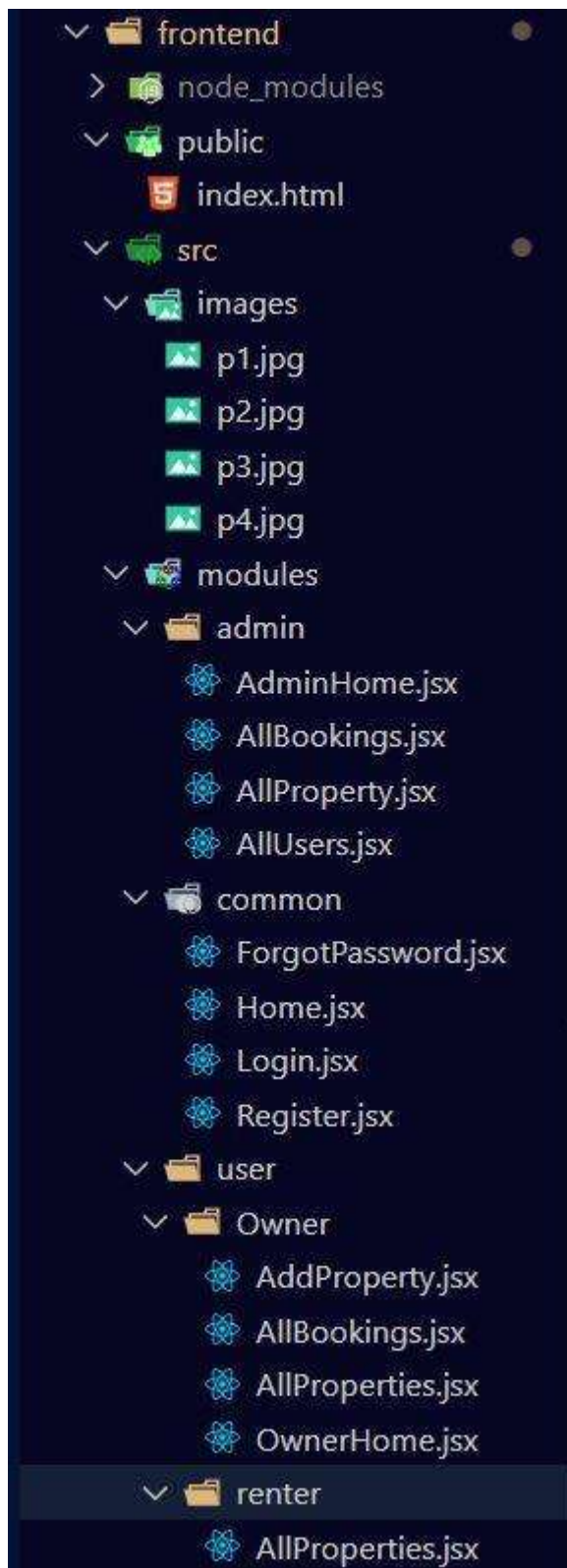
- To start the development server, execute the following command:

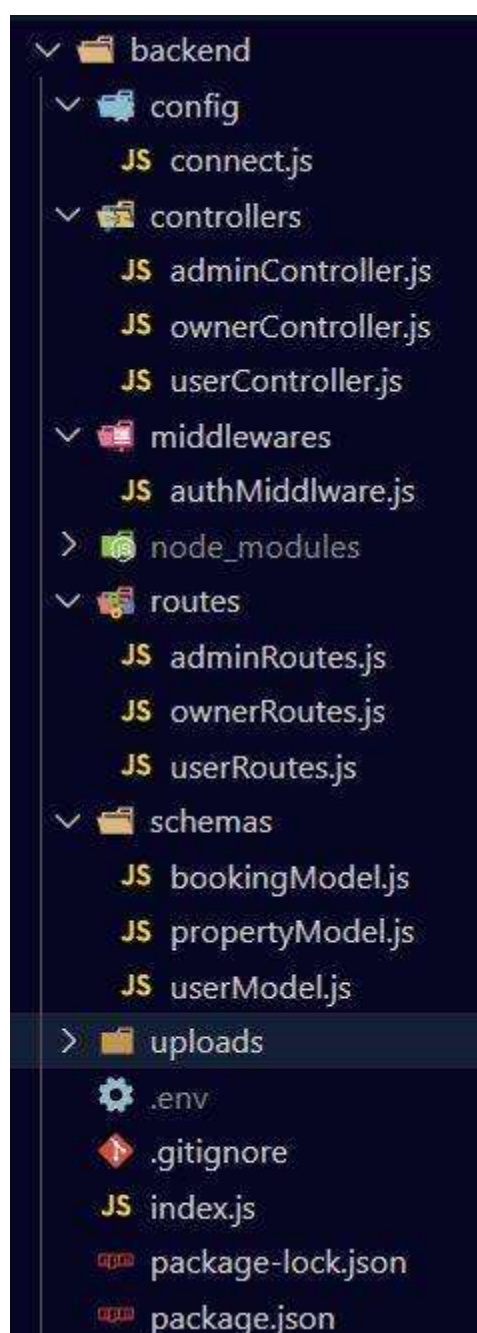
```
npm start
```

- 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:





The first image is of frontend part which is showing all the files and folders that have been used in UI development

The second image is of Backend part which is showing all the files and folders that have been used in backend development.

Application Flow Roles and Responsibilities:

The project has 2 type of user – Renter and Owner and other will be Admin which takes care to all the user. The roles and responsibilities of these two types of users can be inferred from the API endpoints defined in the code. Here is a summary:

Renter/Tenant:

Create an account and log in to the system using their email and password.

They will be shown automatically all the properties in their dashboard.

After clicking on the Get Info, all the information of the property and owner will come and small form will generate in which the renter needs to send his/her details.

After that they can see their booking in booking section where the status of booking will be showing “pending”. It will be change by owner of the property.

Admin:

He/she can approve the user as “owner” for the legit user to add properties in his app

He monitors the applicant of all doctors and approve them and then doctors are registered in the app.

Implement and enforce platform policies, terms of service, and privacy regulations.

Owner:

Gets the approval from the admin for his Owner account.

After approval, he/she can do all CRUD operation of the property in his/her account

He/she can change the status and availability of the property.

Project Setup And Configuration Folder setup:

Create frontend and

Backend folders

2. Open the backend folder to install necessary tools

For backend, we use:

cors

bcryptjs

express

dotenv

mongoose

Moment

Multer

Nodemon

jsonwebtoken

Backend Development

Setup express server

Create index.js file in the server (backend folder).

define port number, mongodb connection string and JWT key in env file to access it.

Configure the server by adding cors, body-parser.

Add authentication: for this,

You need to make middleware folder and in that make authMiddleware.js file for the authentication of the projects and can use in.

Database Development

Configure MongoDB

Import mongoose.

Add database connection from config.js file present in config folder

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;

```

Frontend Development

Installation of required tools:

For frontend, we use:

React

Bootstrap

Material UI

Axios

Moment

Antd

mdb-react-ui-kit

react-bootstrap


Project Implementation & Execution

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 one's provided below

Register or Sign Up:

HOUSEHUNT

HomeLoginRegister



Sign up

Renter Full Name/Owner Name

Email Address

Password

User Type


SIGN UP

Have an account? [Sign In](#)

Login and register page:

HOUSE HUNT

HomeLoginRegister



Sign In

Email Address

Password

SIGN UP

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

Admin Panel:

[ALL PROPERTIES](#) [BOOKING HISTORY](#)

Filter By:

No Properties available at the moment.

Project demo: https://drive.google.com/file/d/1enBJk-X3-ScODu_FMvZRJinwFIDdngb1/view?usp=drive_link