Househunt:Finding Your Perfect Rental Home

INTRODUCTION

A house rent app is typically a mobile or web application designed to help users find rental properties, apartments, or houses for rent. These apps often offer features to make the process of searching for and renting a property more convenient and efficient. Here are some common features you might find in a house rent app:

DESCRIPTION

HouseHunt is a modern MERN-based (MongoDB, Express.js, React, Node.js) rental platform designed to simplify the process of finding and managing rental homes. Whether you're a renter searching for your next apartment or a property owner looking to list and manage properties, HouseHunt streamlines the entire experience.

The app offers a rich set of features:

- **User Registration & Roles:** Renters and Owners can create accounts tailored to their needs, while Admins oversee platform governance.
- **Property Listings & Advanced Search:** Detailed property listings with photos, descriptions, and rental terms, coupled with powerful search filters for location, price, property type, and more.
- **Inquiry & Booking:** Renters can easily inquire about properties and send booking requests to owners directly through the app.
- **Owner & Property Management:** Owners manage their property listings, update availability, and respond to rental requests. Admins verify owner credentials and maintain platform integrity.
- **Lease & Transactions:** In-app messaging helps renters and owners finalize lease agreements, ensuring transparent communication and secure transactions.
- **Notifications & Status Tracking:** Users receive real-time updates on bookings, approvals, and other important actions.

Technical Highlights:

- **Frontend:** Built with React, leveraging Bootstrap and Material UI for a responsive, intuitive design. Axios facilitates smooth communication with backend services.

- **Backend:** Node.js and Express.js manage server-side logic and REST APIs, handling everything from user authentication to booking workflows.

- **Database:** MongoDB provides a scalable, flexible data store for user profiles, property details, and booking transactions.

- **Security & Governance:** Admins monitor activity, verify owners, and enforce policies to maintain a safe, trustworthy ecosystem.

From browsing and booking a new apartment to managing property listings and lease agreements, HouseHunt offers an end-to-end solution that's secure, efficient, and user-friendly.

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.

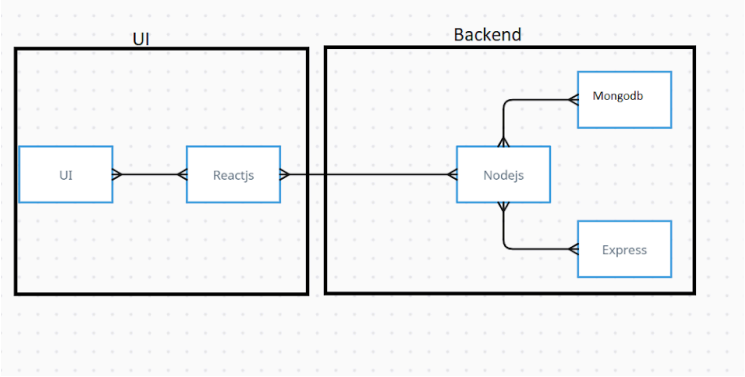
Transaction and Lease Agreement: Once Alice's booking is confirmed, she and the owner negotiate the terms of the lease agreement through the app's messaging system.

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

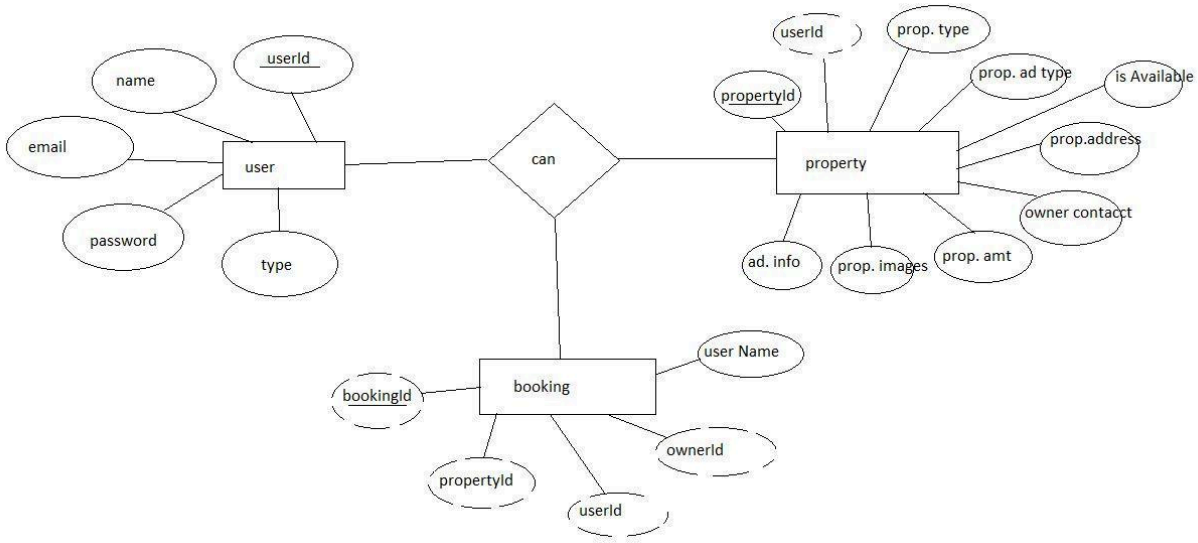
only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful Apis.

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



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/

```
npm init
```

?**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/

?**Moment.js:**

Momentjs is a JavaScript package that makes it simple to parse, validate, manipulate, and display date/time in JavaScript. Moment. js allows you to display dates in a human-readable format based on your location. Install React.js, a JavaScript library for building user interfaces.
Follow the installation guide: https://momentjs.com/

?**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

?**Antd:**

Ant Design is a React. js UI library that contains easy-to-use components that are useful for building interactive user interfaces. It is very easy to use as well as integrate. It is one of the smart options to design web applications using react.
Follow the installation guide: https://ant.design/docs/react/introduce

?**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 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 boostrap.

Install Dependencies:

• Navigate into the cloned repository directory:

```
cd house-rent
```

• Install the required dependencies by running the following commands:

```
cd frontend
npm install
cd ../backend
npm install
```

Start the Development Server:

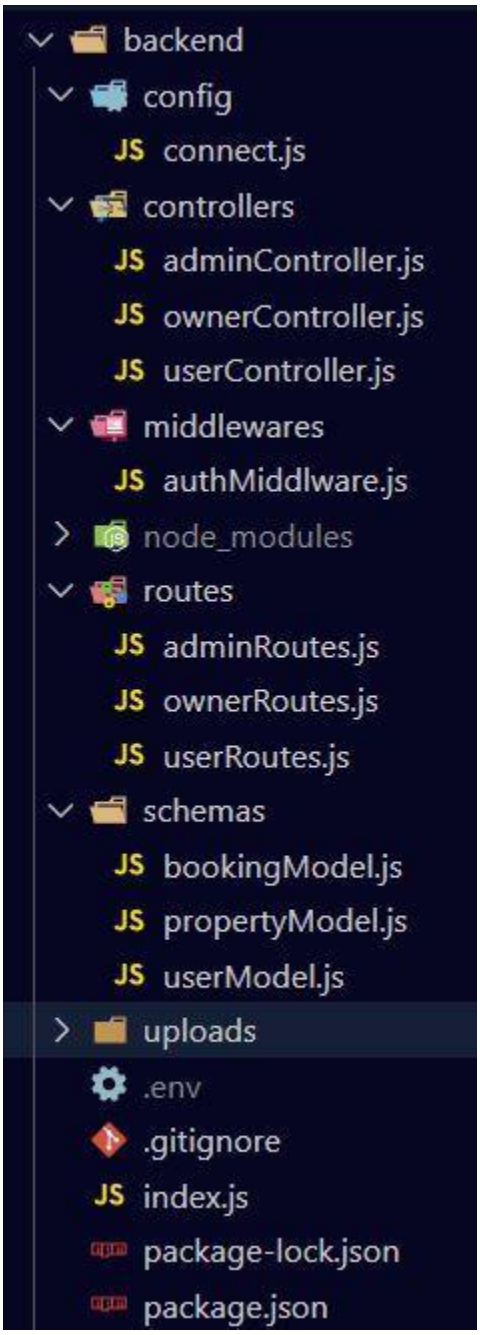
• To start the development server, execute the following command:

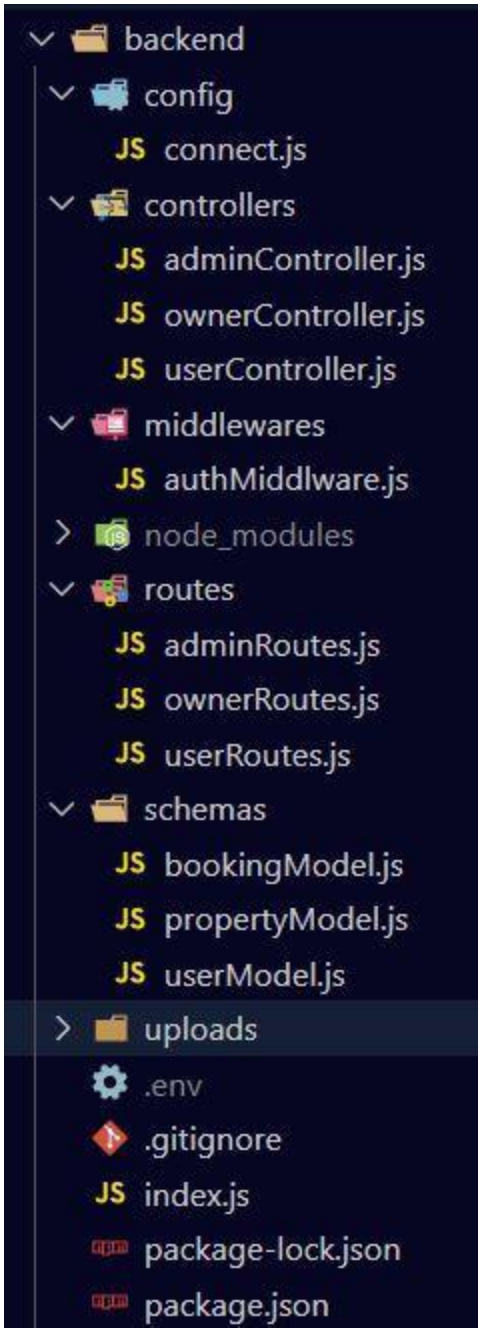
```
npm start
```

• The house rent 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/Tenent:

1. Create an account and log in to the system using their email and password.
2. They will be shown automatically all the properties in their dashboard.
3. 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.
4. 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:

5. He/she can approve the user as “owner” for the legit user to add properties in his app
6. He monitors the applicant of all doctors and approve them and then doctors are registered in the app.
7. Implement and enforce platform policies, terms of service, and privacy regulations.

Owner:

8. Gets the approval from the admin for his Owner account.
9. After approval, he/she can do all CRUD operation of the property in his/her account
10. He/she can change the status and availability of the property.

Project Flow:

Before starting to work on this project, let’s see the demo.
Project demo:
https://drive.google.com/file/d/1rViUmBUaCpJgp7yjL0AkMFK1tCliWsJS/view?usp=sharing
Use the code at: https://mahesh-7989.github.io/Househunt-Finding-Your-Perfect-Rental-Home/
or follow the videos below for a better understanding.

Project Setup And Configuration

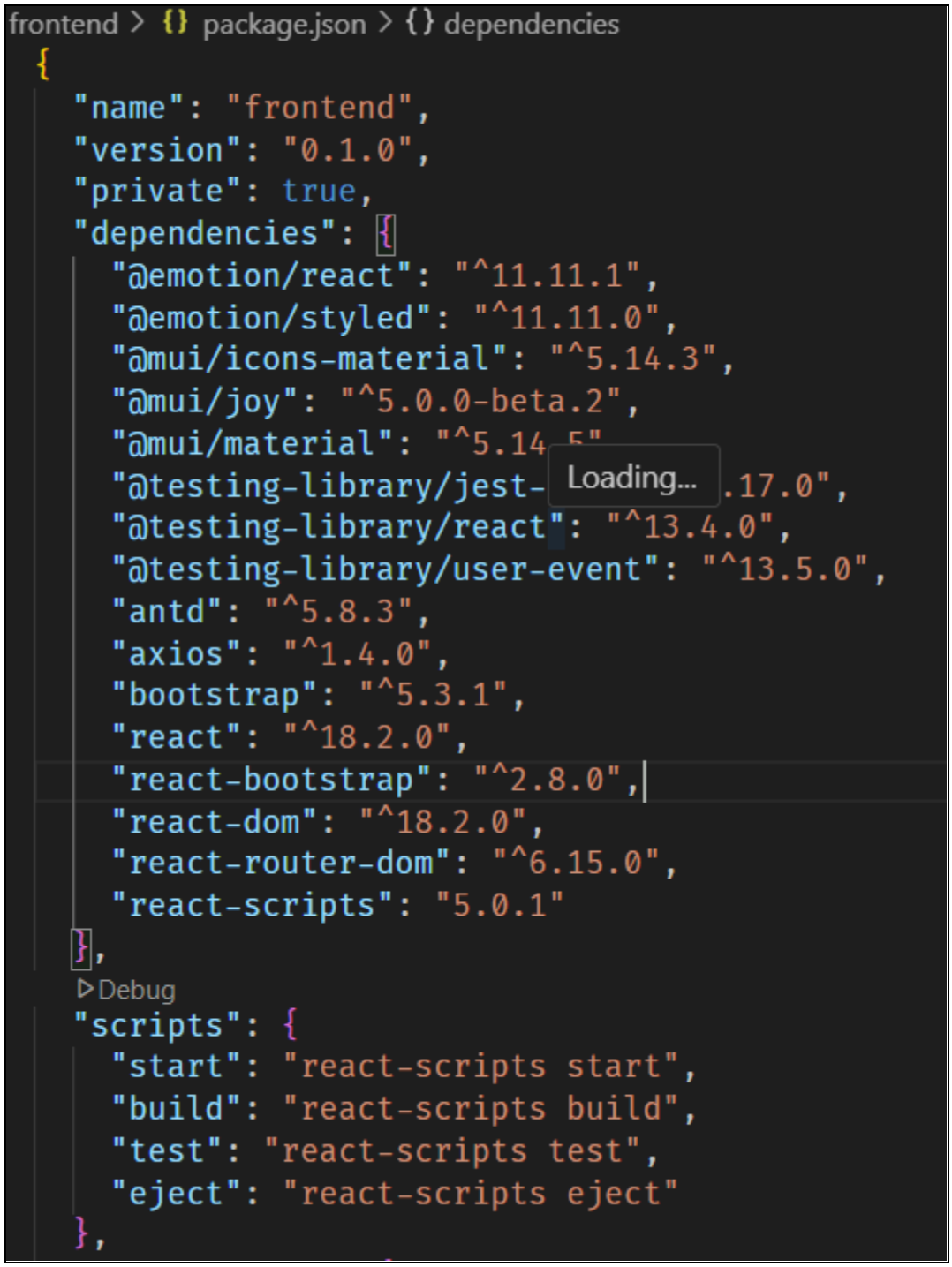
- Folder setup:

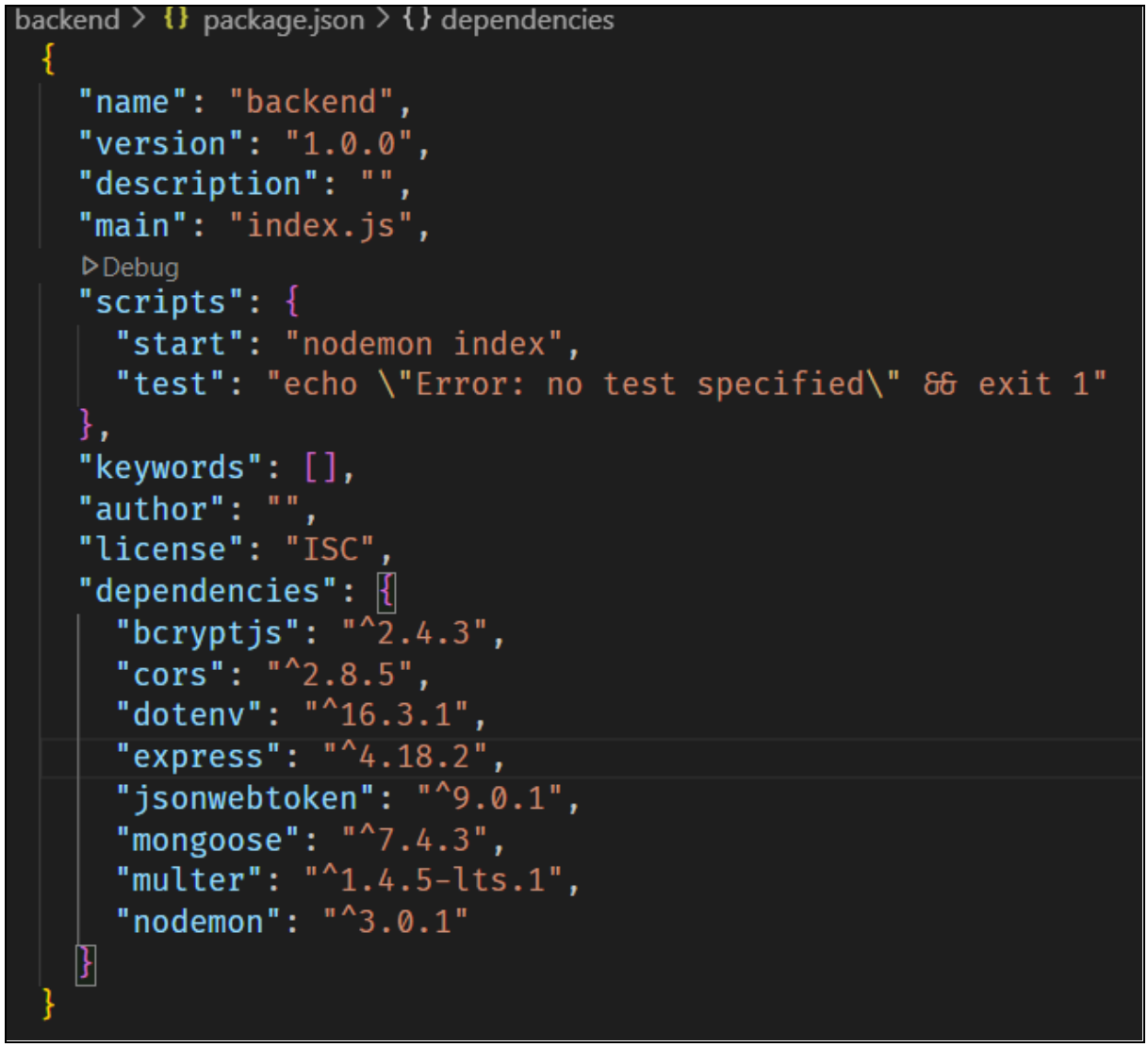
1. Create frontend and
2. Backend folders

2. Open the backend folder to install necessary tools

For backend, we use:

- cors
- bcryptjs
- express
- dotenv
- mongoose
- Moment
- Multer
- Nodemon
- jsonwebtoken


frontend > {} package.json > {} dependencies
{
"name": "frontend",
"version": "0.1.0",
"private": true,
"dependencies": {
"@emotion/react": "^11.11.1",
"@emotion/styled": "^11.11.0",
"@mui/icons-material": "^5.14.3",
"@mui/joy": "^5.0.0-beta.2",
"@mui/material": "^5.14.5"
Loading...
"@testing-library/jest-dom": "^5.17.0",
"@testing-library/react": "^13.4.0",
"@testing-library/user-event": "^13.5.0",
"antd": "^5.8.3",
"axios": "^1.4.0",
"bootstrap": "^5.3.1",
"react": "^18.2.0",
"react-bootstrap": "^2.8.0",
"react-dom": "^18.2.0",
"react-router-dom": "^6.15.0",
"react-scripts": "5.0.1"
},
Debug
"scripts": {
"start": "react-scripts start",
"build": "react-scripts build",
"test": "react-scripts test",
"eject": "react-scripts eject"
},



Backend Development

- **Setup express server**

1. Create index.js file in the server (backend folder).
2. define port number, mongodb connection string and JWT key in env file to access it.
3. Configure the server by adding cors, body-parser.

- **Add authentication:** for this,

1. 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

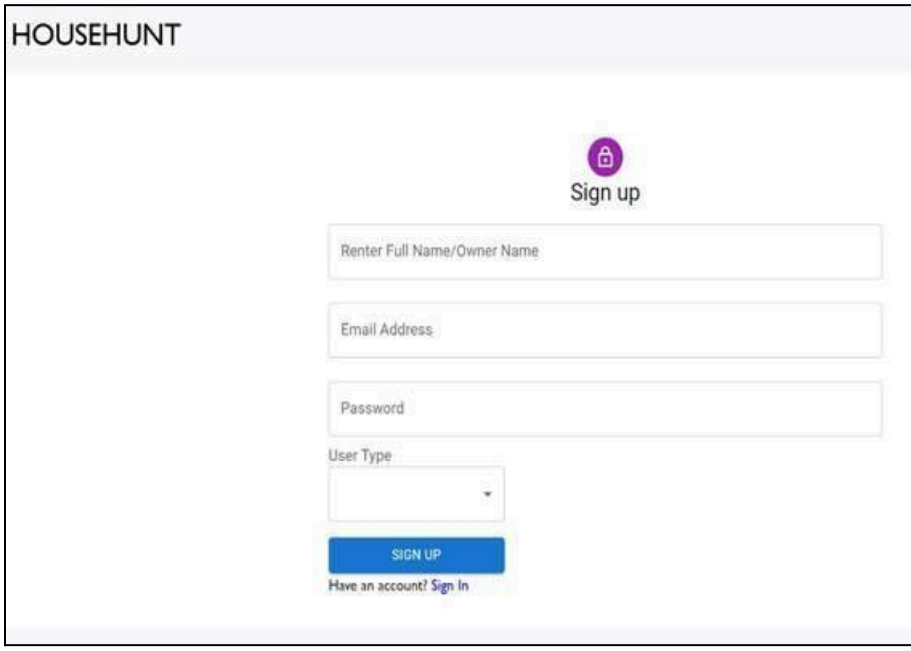
- **Installation of required tools:**
- For frontend, we use:

1. React
2. Bootstrap
3. Material UI
4. Axios
5. Moment
6. Antd
7. mdb-react-ui-kit
8. react-bootstrap

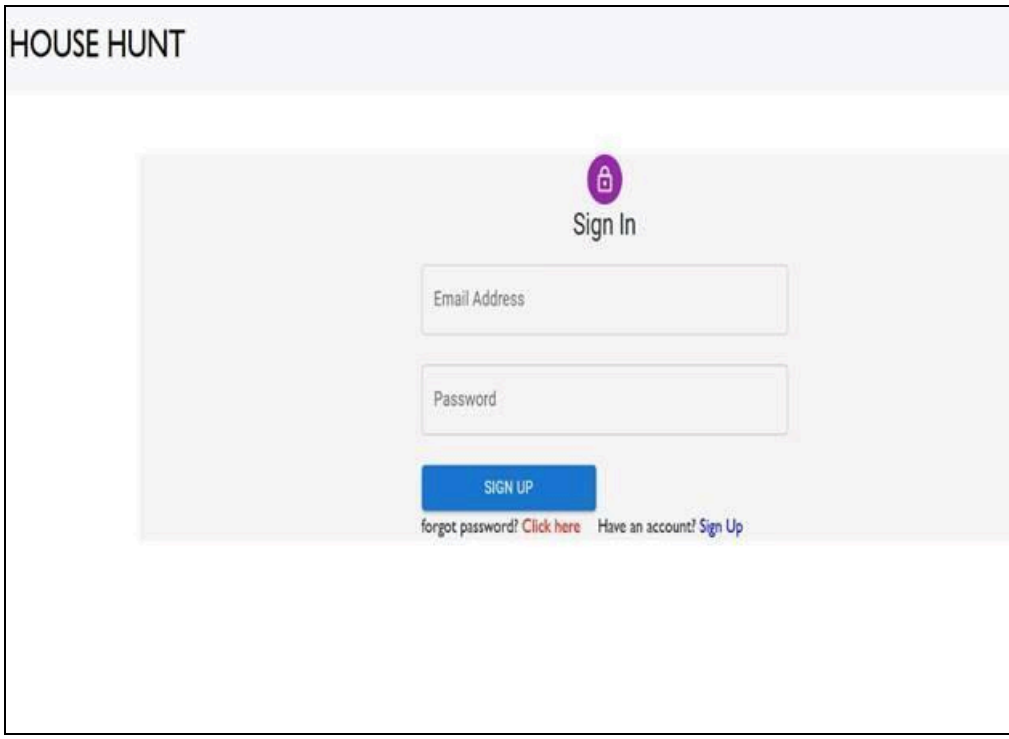
Project Implementation & Execution

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 belo

Register or Sign Up:



Login and register page:

- 

Admin Panel:

- 

Project demo:

https://drive.google.com/file/d/1rViUmBUaCpJgp7yjL0AkMFK1tCliWsJS/view?usp=sharing