



**Project Title:** perfect Rental Home

**Team Members:**

**Team Leader : Muttireddy Rajgopal**

**Team member : Vegesana Sairam**

**Team member : Molli Gowtham Kumar**

# Project Overview

## Purpose:

Finding your perfect rental home allows you to discover a living space that meets your needs, budget, and lifestyle. It provides a comfortable and secure environment, promoting well-being and happiness. A perfect rental home also offers convenience, amenities, and a sense of community.

## Features:

- **Personalized Search**
- **Detailed Listings**
- **Accurate Location**
- **Amenity Filters**
- **Reviews and Ratings**

## HouseHunt: Finding Your Perfect Rental Home

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:

**Property Listings:** The app provides a database of available rental properties, complete with detailed descriptions, photos, location, rent amount, and other relevant information.

**Search Filters:** Users can apply various filters to narrow down their search results based on criteria such as location, rent range, property type (apartment, house, room, etc.), number of bedrooms, amenities, and more.

**Contact Landlords/Property Managers:** The app might provide a way for users to contact the property owners or managers directly through the app, often through messaging or email.

### 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.

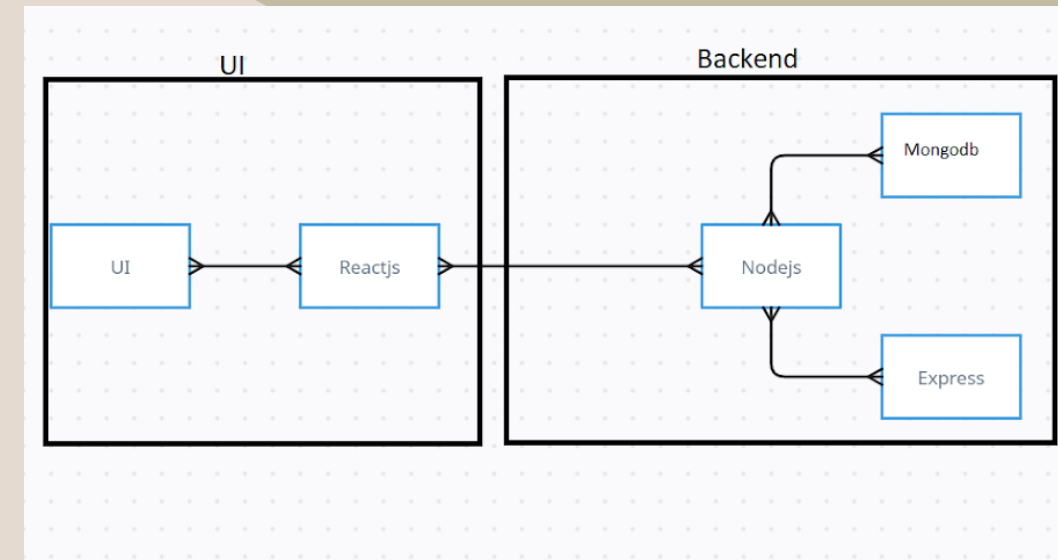
The technical architecture of our House rent app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using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

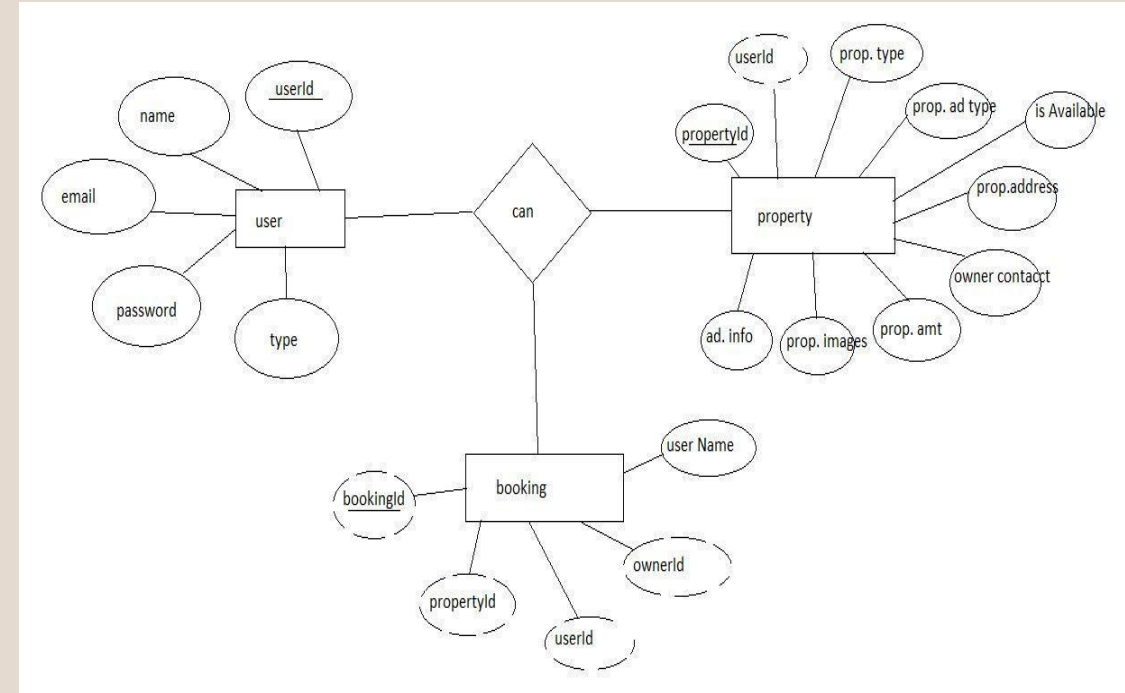
Here there is 3 collections namely users, property, and booking which have their own fields in

Users:

- 1.\_id: (MongoDB creates by unique default)
- 2.name
- 3.email
- 4.password
- 5.type

Property:

- 1.userID: (can be act as foreign key )
- 2.\_id: (MongoDB creates by unique default)
- 3.prop.Type
- 4.prop.AdType
- 5.isAvailable
- 6.prop.Address
- 7.owner contact
- 8.prop.Amt
- 9.prop.images
- 10.add.Info



# 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.



**?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 bootstrap.

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>

# 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:

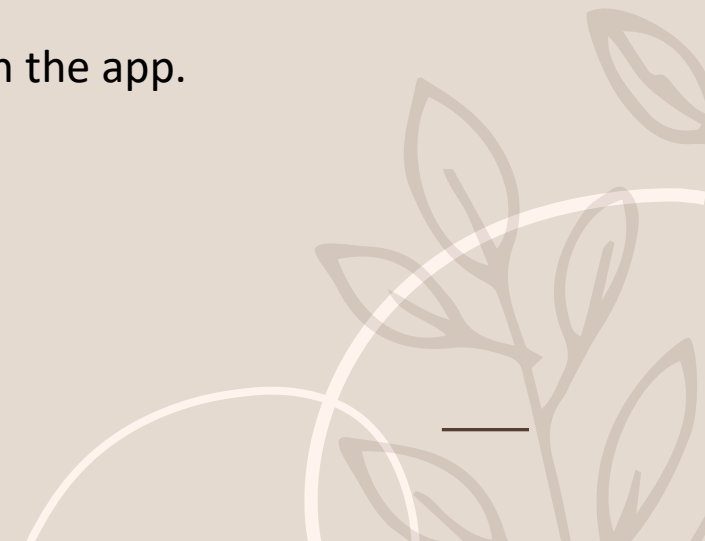
- 1.Create an account and log in to the system using their email and password.
- 2.They will be shown automatically all the properties in their dashboard.
- 3.After clicking on the Get Info, all the information of the property and owner will come and small form will generate in which the renter needs 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:

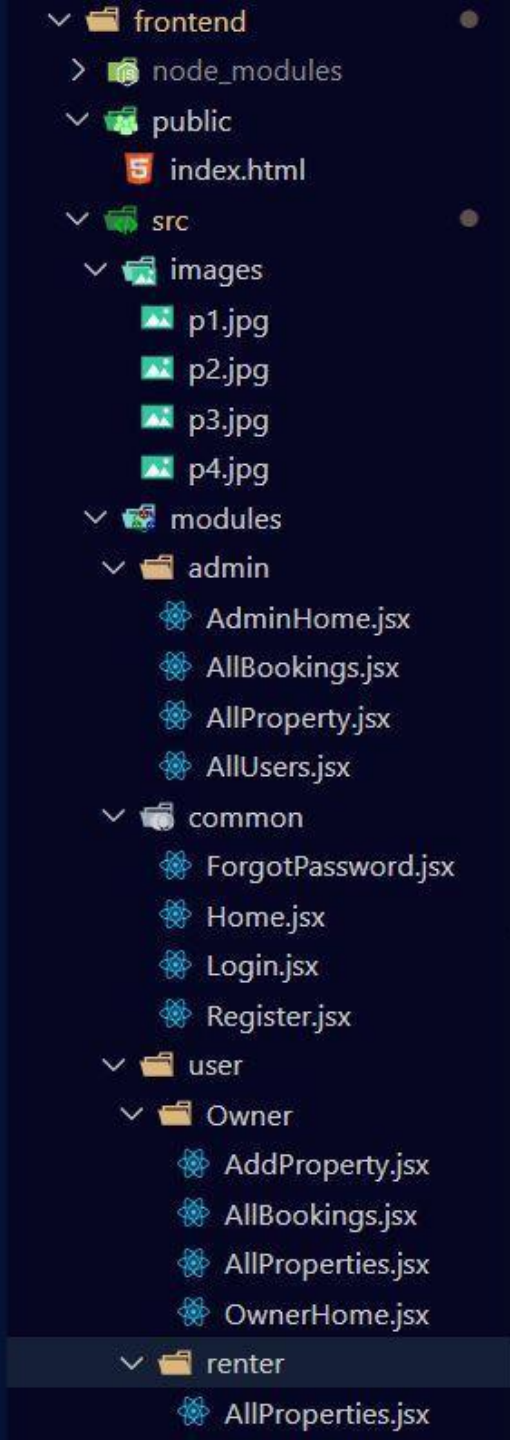
- 5.He/she can approve the user as “owner” for the legit user to add properties in his app
- 6.He monitors the applicant of all doctors and approve them and then doctors are registered in the app.
- 7.Implement and enforce platform policies, terms of service, and privacy regulations.

### 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.

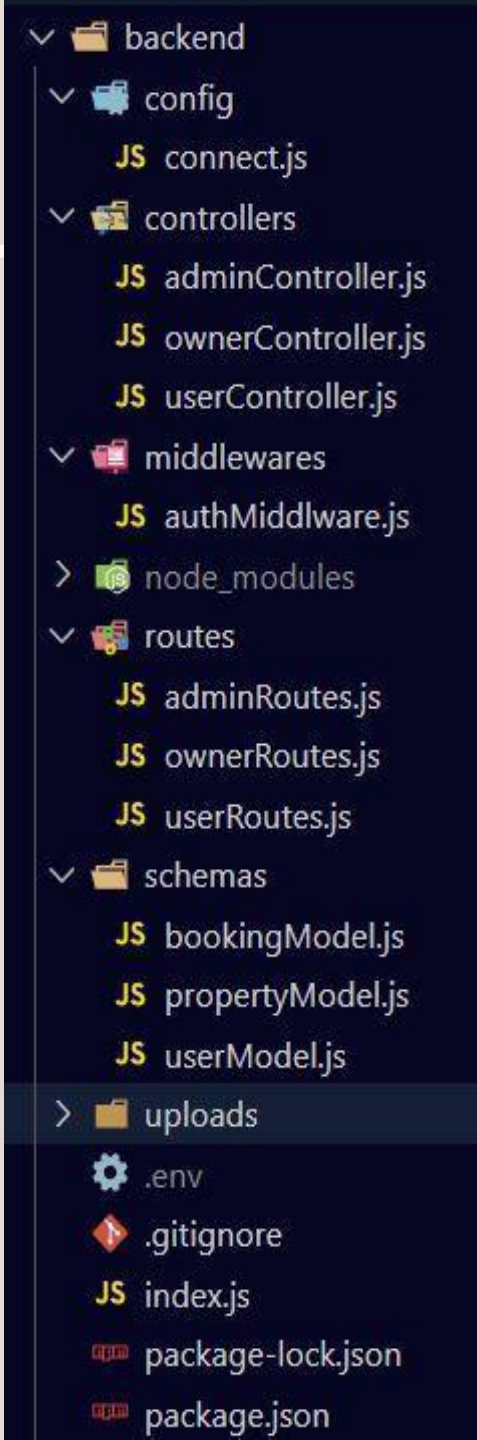






The first image is of frontend part which is showing all the files and folders that have been used in UI development

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



# 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",
    "@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"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
}
```

backend > {} package.json > {} dependencies

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon index",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.1",
    "mongoose": "^7.4.3",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^3.0.1"
  }
}
```

## 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

- 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;
```

## Frontend development

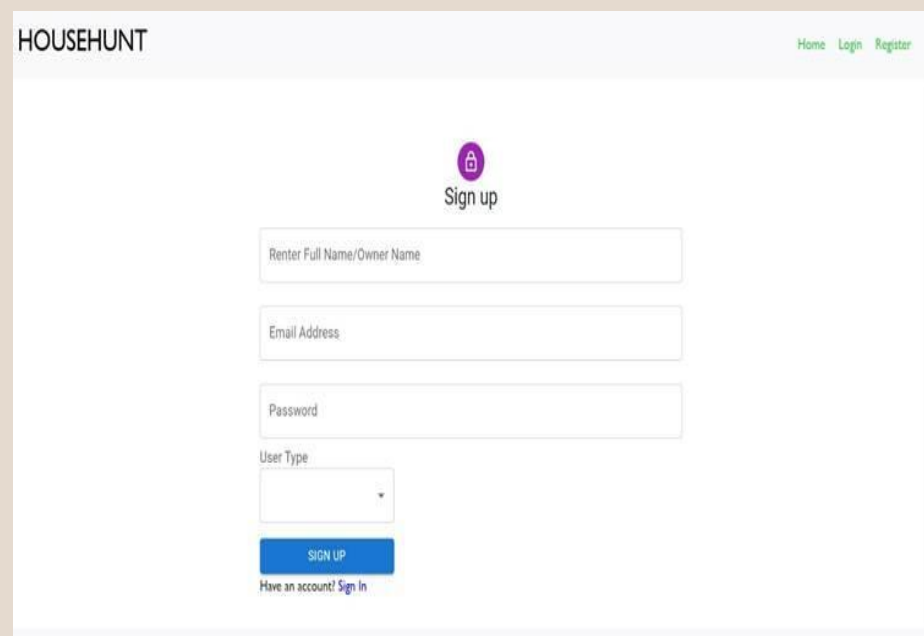
### •Installation of required tools:

•For frontend, we use:

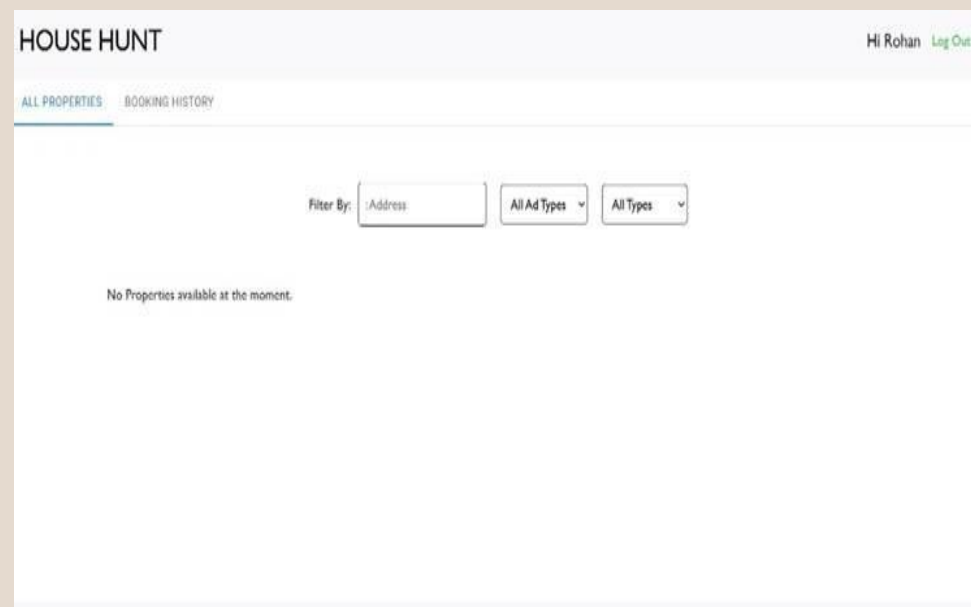
- 1.React
- 2.Bootstrap
- 3.Material UI
- 4.Axios
- 5.Moment
- 6.Antd
- 7.mdb-react-ui-kit
- 8.react-bootstrap

## 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 below



The screenshot shows the 'Sign up' page of the 'HOUSEHUNT' application. The header includes the 'HOUSEHUNT' logo on the left and 'Home', 'Login', and 'Register' links on the right. The main content area features a purple lock icon and the text 'Sign up'. Below this, there are four input fields: 'Renter Full Name/Owner Name', 'Email Address', 'Password', and 'User Type' (a dropdown menu). A blue 'SIGN UP' button is positioned below the fields. At the bottom, there is a link that says 'Have an account? Sign In'.



The screenshot shows the 'HOUSE HUNT' user interface for a logged-in user named 'Hi Rohan'. The header includes the 'HOUSE HUNT' logo on the left and 'Hi Rohan' and 'Log Out' links on the right. Below the header, there are two tabs: 'ALL PROPERTIES' (active) and 'BOOKING HISTORY'. The main content area displays a 'Filter By:' section with three dropdown menus: 'Address', 'All Ad Types', and 'All Types'. Below the filters, a message states 'No Properties available at the moment.'.



Thank you