

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.

HouseHunt is a full-stack web and mobile application developed using the **MERN stack** (MongoDB, Express.js, React.js, Node.js). It is designed to **bridge the gap between renters, property owners, and administrators**, offering a centralized, secure, and efficient platform for property rental. The app ensures **user-friendly navigation, transparency in property details, and streamlined communication**, reducing the complexity often associated with renting homes in urban areas.

The purpose of HouseHunt is to streamline the real estate journey by providing a comprehensive, user-friendly platform that connects home buyers, sellers, renters, and real estate professionals. It aims to simplify the often complex and time-consuming process of searching for or listing a property, offering tools and resources that enhance the experience for all parties involved.

1. **Empowering Users with Information:** HouseHunt provides detailed listings, market insights, and property data, helping users make informed decisions about buying, selling, or renting homes.
 2. **Simplifying the Property Search:** With advanced search filters and features like virtual tours, HouseHunt makes it easy to find properties that match specific preferences such as location, price, and amenities.
 3. **Facilitating Transactions:** By connecting buyers with sellers, renters with landlords, and users with real estate professionals, HouseHunt streamlines the entire transaction process, from browsing listings to negotiating deals and finalizing paperwork.
 4. **Supporting Real Estate Professionals:** HouseHunt serves as a marketing tool for real estate agents and brokers, offering them a platform to reach potential clients and showcase properties.
 5. **Creating a Transparent Marketplace:** By providing clear, accurate property information, HouseHunt fosters transparency, ensuring that users have access to reliable data to make confident real estate decisions.
- Offering a Convenient, 24/7 Platform:** HouseHunt is accessible anytime, anywhere, offering user

2.Features Of House Hunt :

HouseHunt offers a variety of features designed to enhance the real estate experience for buyers, sellers, renters, and professionals. These features aim to simplify property searches, provide valuable insights, and streamline the transaction process. Below are some of the key features of HouseHunt:

✓ Property Listings

HouseHunt provides a **dynamic and easily searchable catalog of available rental properties**, offering:

- Location with map preview
- Monthly rent, security deposit details
- Property type (apartment, shared room, house, etc.)
- Full image gallery with carousel support
- Amenities (Wi-Fi, AC, furnished, pet-friendly, etc.)
- Availability status and date of posting

✓ Search Filters

To personalize the property search, HouseHunt includes **multi-level filtering**, enabling users to refine search results based on:

- City or ZIP code
- Budget range (min-max rent)
- Number of bedrooms/bathrooms
- Property type
- Essential amenities (e.g., elevator, parking, water supply, etc.)

✓ Contacting Owners

Users can **interact directly with property owners** via:

- In-app messaging system (text-based chat)
- Inquiry submission form for each listing
- Email notifications for message responses
- Owner contact panel (available post-inquiry approval)

3. Scenario-Based Case Study: Renting an Apartment

Alice (Renter)

- Registers on the platform using a secure email/password combination.
- Explores the homepage with featured and recommended properties.
- Applies filters to narrow down her ideal apartment.
- Views complete listing details, amenities, and owner profile.
- Submits an inquiry and waits for approval.

Bob (Owner)

- Signs up as a property owner, submits identity verification documents.
- Waits for approval by admin before listing properties.
- Adds new property with photos, rent, availability, and amenities.
- Manages multiple listings and responds to tenant inquiries via the message dashboard.

Admin

- Verifies new property owners to prevent fraudulent listings.
- Approves or rejects listing submissions based on quality and policy.
- Monitors user activity, manages complaints, and enforces rules.

Transaction Flow

- Alice submits an inquiry for Bob's apartment.
- Bob reviews Alice's profile and approves the inquiry.
- Alice receives a notification and listing status updates to "Pending Owner Confirmation."
- Alice and Bob negotiate the rental terms using the integrated messaging tool.
- Lease agreement is finalized via a downloadable document or offline meeting.
- Alice confirms move-in and transaction is marked as complete.

Depth:

1. Advanced Property Search

Filters: Users can filter properties based on location, price range, property type, size, number of bedrooms, and specific amenities (e.g., pet-friendly, pool, garage).

Saved Searches: Allows users to save their search criteria for easy access to new listings that match their preferences.

2. Interactive Property Listings

High-Quality Photos & Videos: Properties are showcased with professional images and video tours to give users a detailed view of the home.

360° Virtual Tours: Users can take interactive virtual tours of properties to get a feel for the space before scheduling an in-person visit.

Floor Plans: Many listings include detailed floor plans to help users visualize the layout of a property.

3. Market Insights and Trends

Price Trends: Displays information on how property prices are trending in specific areas, allowing buyers and sellers to make informed decisions.

Neighbourhood Insights: Information on local amenities, schools, crime rates, and transport options, giving users a deeper understanding of the area they are considering.

Investment Potential: Provides data on property value appreciation and investment opportunities for users looking to invest in real estate.

4. Property Alerts

New Listings Alerts: Users can set up email or push notifications to be alerted when new properties matching their criteria are listed.

Price Drop Alerts: Users are notified when a property they are interested in has a price reduction.

5. Professional Connection

Real Estate Agents and Brokers: HouseHunt connects users with certified real estate professionals who can guide them through the buying, selling, or renting process.

Legal and Financial Experts: The platform offers connections to legal advisors, mortgage brokers, and financial planners to assist with the paperwork and financial aspects of the transaction.

4. Technical Architecture

◆ Client-Server Model

- **Frontend (React.js):** Uses reusable components, Redux for state management, React Router for navigation.
- **Styling:** Bootstrap for responsive design and Material UI for modern UI elements.

◆ Backend Services

- **Node.js & Express.js:** Build RESTful API endpoints for user actions (CRUD operations).
- **MongoDB:** NoSQL database used to manage structured and semi-structured data.
- **Moment.js:** Handles time stamps and formats for inquiries, messages, and transactions

◆ API Modules

Key RESTful APIs:

/api/users – registration, login, user roles

/api/properties – create, update, fetch listings

/api/inquiries – send/receive contact requests

/api/messages – chat functionality






5. Data Models / MongoDB Collections

Collection	Purpose
Users	Stores information for renters, owners, and admins. Includes role, email, password (hashed), and profile data.
Properties	Contains all property listings, owner reference, availability status, location, rent, amenities.
Inquiries	Links renters to properties with status (pending, approved, rejected).
Messages	Stores chat messages, timestamps, and references to users and inquiries.
Approvals	Maintains records of owner verification and property approvals by admins.

6. Platform Governance & Security

- **Role-Based Access Control (RBAC):**
 - **Renter:** Can view and inquire about listings.
 - **Owner:** Can list properties and respond to inquiries.
 - **Admin:** Full access to monitor and moderate the platform.
- **Authentication & Data Security:**
 - Passwords stored securely using hashing.
 - JWT (JSON Web Tokens) used for session management and authorization.
- **Admin Panel Tools:**
 - Approve/reject owner registrations and listings.
 - View flagged content or inappropriate listings.
 - Audit logs for monitoring actions taken by any user role.
- **Privacy & Safety Policies:**
 - Users can report suspicious listings or users.
 - Communication is monitored for abuse or spam using filters.

7. Future Enhancements

-  **Online Payments:**
 - UPI, Razorpay, Stripe integration for secure rent payments and deposits.
 -  **Live Chat Support:**
 - Real-time communication using WebSockets or Firebase Realtime DB.
 -  **AI Recommendations:**
 - Personalized property suggestions using user behavior, preferences, and search history.
 -  **Ratings & Reviews:**
 - Feedback system to rate owners, renters, and properties.
 -  **Map View:**
 - Google Maps integration for geographic property browsing and location previews.
-

9. Advanced Modules & Features

1. Smart Notifications

- Real-time alerts via email or push notifications:
 - New message received
 - Inquiry status changed
 - New property listed in user's preferred location
 - Owner approval/rejection updates

2. Favorites & Wishlist

- Users can **bookmark or save properties** to revisit later.
- "Favorites" section allows comparison of shortlisted homes.

3. Analytics Dashboard (Admin/Owner)

- Owners see metrics like:
 - Number of views per property
 - Inquiries received
 - Average response time
- Admin gets system-wide stats:
 - User growth
 - Daily/weekly listing trends
 - Most searched areas

4. Blog/Resource Section

- Tips for renters (legal checks, documents needed)
- Owner guidelines (how to photograph property, tenant rights)
- SEO-optimized content to attract new users via search engines

5. Document Upload System

- Owners can upload:
 - Rental agreements
 - Property verification documents
- Renters can upload:
 - ID proof
 - Salary slips / student ID
- Admins verify and store these securely in MongoDB GridFS or AWS S3

6. Multi-Language Support

- Option to switch between English, Hindi, Telugu, etc.
- Useful for users from different regions in India

7. Mobile App Version (React Native / Flutter)

- Syncs with the same backend
- Push notifications
- Optimized for low-data usage



10. Technologies and Tools Used

Category	Tools / Libraries
Frontend	React.js, React Router, Redux, Axios
Styling	Bootstrap, Material UI, CSS Modules
Backend	Node.js, Express.js
Database	MongoDB, Mongoose ODM
Authentication	JWT, bcrypt.js
Real-time Features	Socket.IO or Firebase (for live chat)
Payment Integration	Razorpay, Stripe (future enhancement)
Hosting	Vercel (Frontend), Render/Heroku (Backend), MongoDB Atlas
Dev Tools	Postman, GitHub, VSCode, Chrome DevTools



11. Deployment and CI/CD

- Frontend and backend are **deployed separately** for modular scaling.
 - Continuous Deployment using:
 - GitHub Actions
 - Webhooks for automatic deploy on code push
 - Environment variables handled securely using `.env` files and cloud secret managers
-

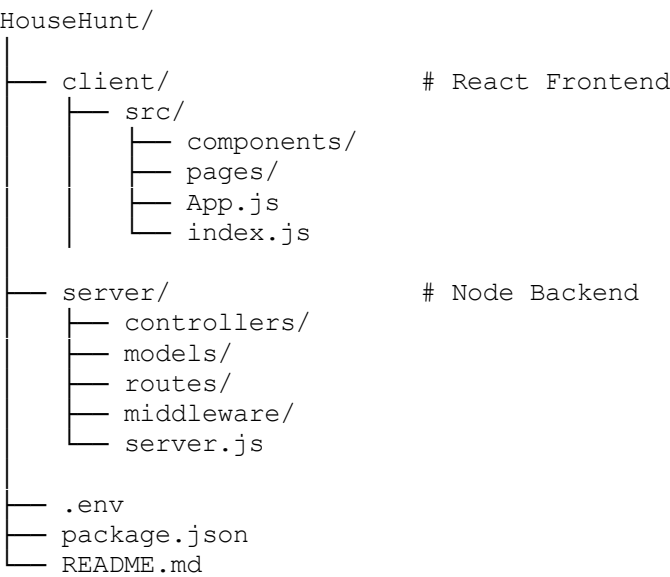
12. Testing Strategy

- **Unit Testing:** With Jest for both frontend and backend functi
- **Integration Testing:** API route validation using Postman/Newman.
- **User Acceptance Testing (UAT):** Manual testing to simulate user journeys.

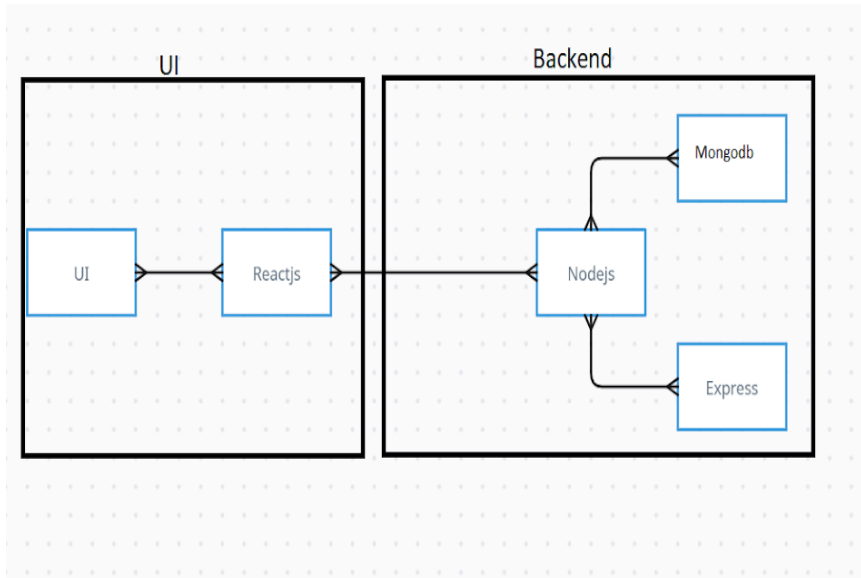
13. User Roles Breakdown

Role	Permissions
Renter	Browse listings, send inquiries, chat with owners, mark favorites
Owner	Add/edit/delete listings, respond to inquiries, view dashboard metrics
Admin	Approve owners/listings, remove inappropriate content, view analytics

14. Folder Structure (Sample)



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.

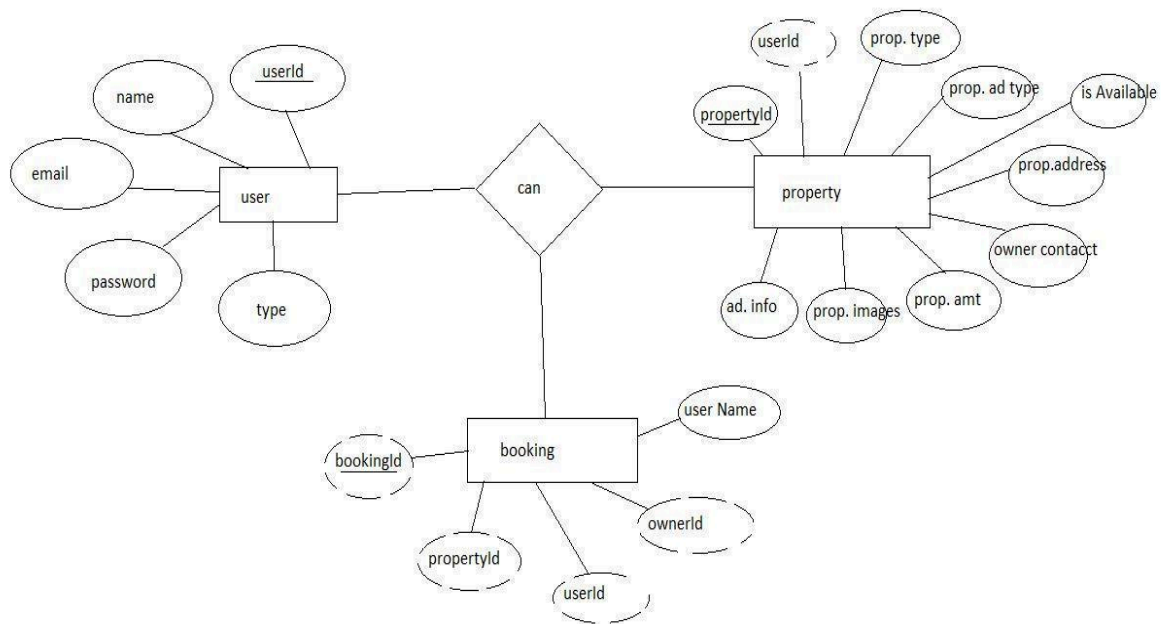
The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user whether it is admin, doctor and ordinary user working on it.

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

ER DAIGRAM



PREREQUISITE:

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:

Moment Js 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 the 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>

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/house-rent.git>

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.

Roles and Responsibilities:

The project has 2 types of users – Renter and Owner and the other will be Admin which takes care of all the users. 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 a small form will be generated in which the renter needs to send his\her details.
4. After that they can see their booking in the booking section where the status of booking will be showing "pending". It will be changed by the owner of the property.

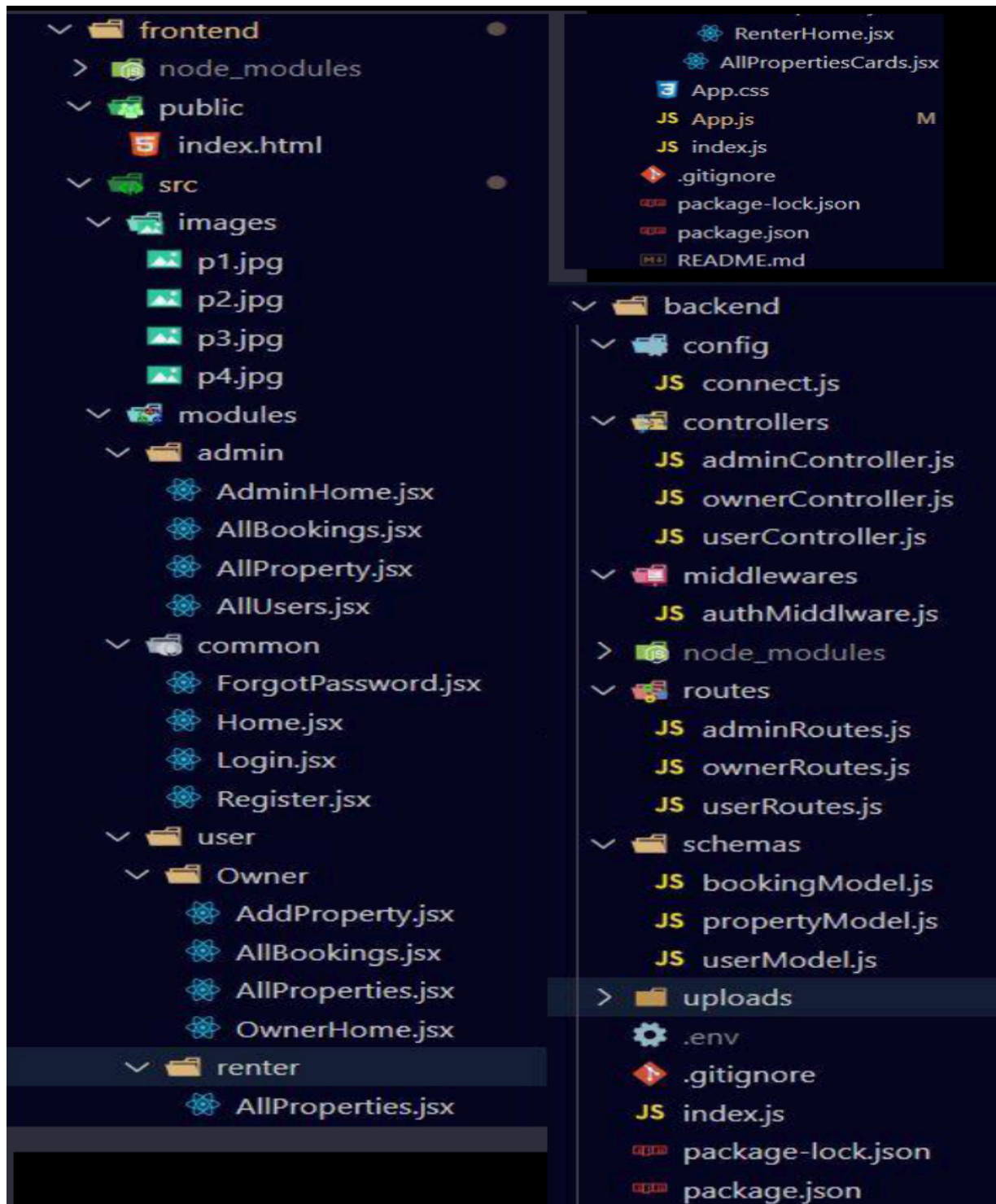
Admin:

1. He/she can approve the user as "owner" for the legit user to add properties in his app
2. He monitors the applicant of all doctors and approves them and then doctors are registered in the app.
3. Implement and enforce platform policies, terms of service, and privacy regulations.

Owner:

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

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

Project Flow:

Before starting to work on this project, let's see the demo.

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

Use the code in: <https://github.com/awdhesh-student/house-rent.git>

or follow the videos below for better understanding.

Milestone 1: Project setup and configuration.

Folder setup:

Create frontend and

Backend folders

Installation of required tools:

1. Open the frontend folder to install necessary tools

For frontend, we use:

React

Bootstrap

Material UI

Axios

Moment

Antd

mdb-react-ui-kit

react-bootstrap

2. Open the backend folder to install necessary tools

For backend, we use:

cors

bcryptjs

express

dotenv

mongoose

Moment

Multer

Nodemon

jsonwebtoken

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

```

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": "Loading... .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"
  },

```

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

```

backend > {} package.json > {} dependencies
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▶ Debug
  "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"
  }
}

```

Milestone 2: 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.

✓ 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, properties schemas.

✓ Add authentication: for this,

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

Milestone 3: Database Development:

- o Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
- o Create a database and define the necessary collections for users, transactions, stocks and orders.
- o Also let's see the detailed description for the schemas used in the database.
- o For the connection of database use the code given below

Milestone 2: 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.

✓ 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, properties schemas.

✓ Add authentication: for this,

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

Milestone 3: Database Development:

- o Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
- o Create a database and define the necessary collections for users, transactions, stocks and orders.
- o Also let's see the detailed description for the schemas used in the database.
- o For the connection of database use the code given below

```
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;
```

Milestone 4: Frontend Development:

Setup React Application:

Bringing SB Stocks to life involves a three-step development process. First, a solid foundation is built using React.js. This includes creating the initial application structure, installing necessary libraries, and organizing the project files for efficient development. Next, the user interface (UI) comes to life. To

start the development process for the frontend, follow the below steps.

Install required libraries.

Create the structure directories.

Design UI components:

Reusable components will be created for all the interactive elements you'll see on screen, from stock listings and charts to buttons and user profiles. Next, we'll implement a layout and styling scheme to define the overall look and feel of the application. This ensures a visually-appealing and intuitive interface. Finally, a navigation system will be integrated, allowing you to effortlessly explore different sections of SB Stocks, like viewing specific stocks or managing your virtual portfolio.

Implement frontend logic:

In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

Integration with API endpoints.

Implement data binding.

5 . Project Implementation :

Register or Sign Up:



Sign up

User Type

Have an account? [Sign In](#)

Sign In

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

Properties:

HOUSE HUNT

Hi Rohan [Log Out](#)

ALL PROPERTIES

BOOKING HISTORY

Booking ID	Property ID	Tenant Name	Phone	Booking Status
------------	-------------	-------------	-------	----------------

HOUSE HUNT

Hi Rohan [Log Out](#)

ALL PROPERTIES

BOOKING HISTORY

Filter By:

All Ad Types

All Types

No Properties available at the moment.

Booking History:

Projectdemo::

https://drive.google.com/file/d/1enBJk-X3-ScODu_FMvZRJinwFI_Ddngb1/view?usp=drive_link

Code:

<https://drive.google.com/drive/folders/10OstrbGEPKtXDENGkerKBShejPJlbVgj?usp=sharing>

Implementation

1. <https://drive.google.com/drive/folders/1hb2OeuCQpGPs5YwrVb2wxINERYeX2bT6>

2. https://drive.google.com/drive/folders/1wDKO3Kfz1ORSE_sLqFHitFSoQsVJi_of