FULL STACK DEVELOPMENT: MERN

1. INTRODUCTION

- **Project Title:**

  House Hunt: Finding your perfect Rental Home

- **Team Members:**

Team ID: LTVIP2025TMID21134

MizraTahseen Raza

Meesala Madhavi

Medaramitla yedukondalu

Md Abdul Hajeem

2. PROJECT OVERVIEW

Purpose:

The purpose of House Hunt 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:** House Hunt 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, House Hunt 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, House Hunt streamlines the entire transaction process, from browsing listings to negotiating deals and finalizing paperwork.

4. **Supporting Real Estate Professionals:** House Hunt 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, House Hunt fosters transparency, ensuring that users have access to reliable data to make confident real estate decisions. Offering a Convenient, 24/7 Platform: House Hunt is accessible anytime, anywhere, offering user

Features Of House Hunt:

House Hunt 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 House Hunt:

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: House Hunt 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.

3. TECHNICAL ARCHITECTURE:

Frontend:

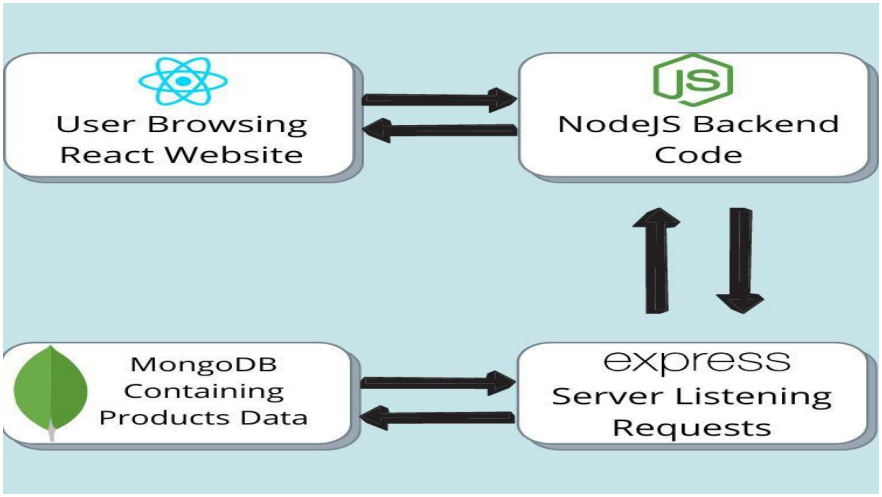
- Built with **React.js**, uses **Bootstrap**, **Material UI**, **Ant Design**, **MDB React UI Kit**.
- Uses **Axios** for API calls.

Backend:

- Developed using **Node.js** and **Express.js**.
- Handles server-side logic and APIs.

Database:

- Uses **MongoDB** for storing data such as users, properties, bookings, etc.
- Uses **Mongoose** for schema modelling.



- 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.

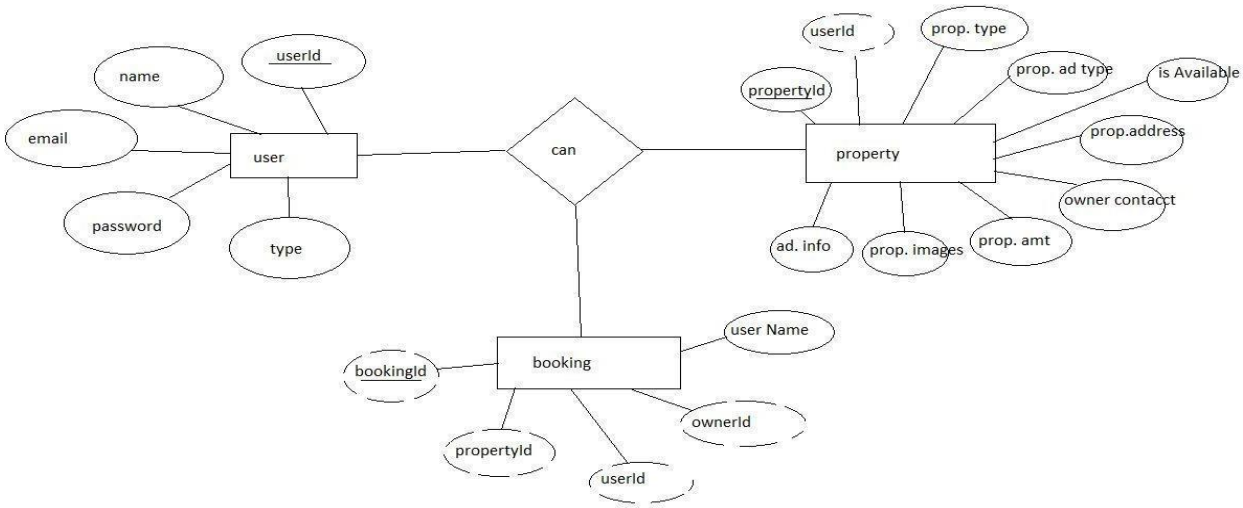
- For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, including user profiles, for booking room, and adding room, etc.

- 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



4. SETUP INSTRUCTIONS

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**

INSTALLATION:

• 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.

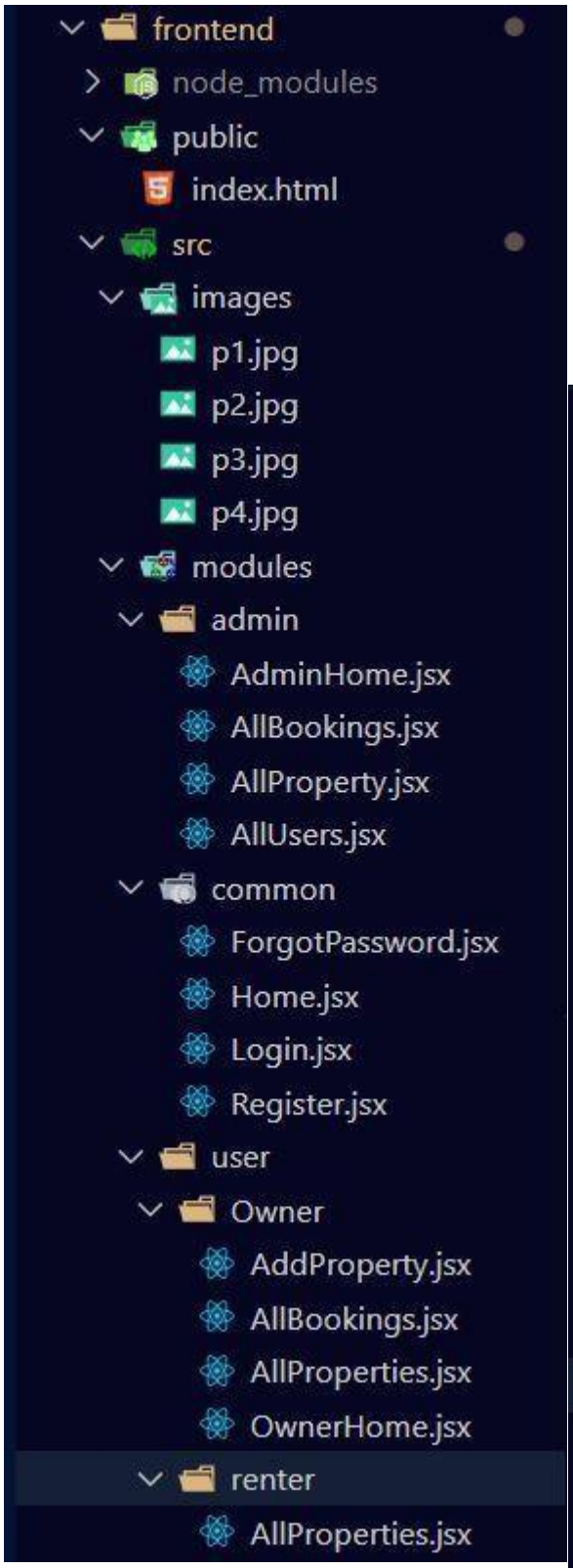
Admin:

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.

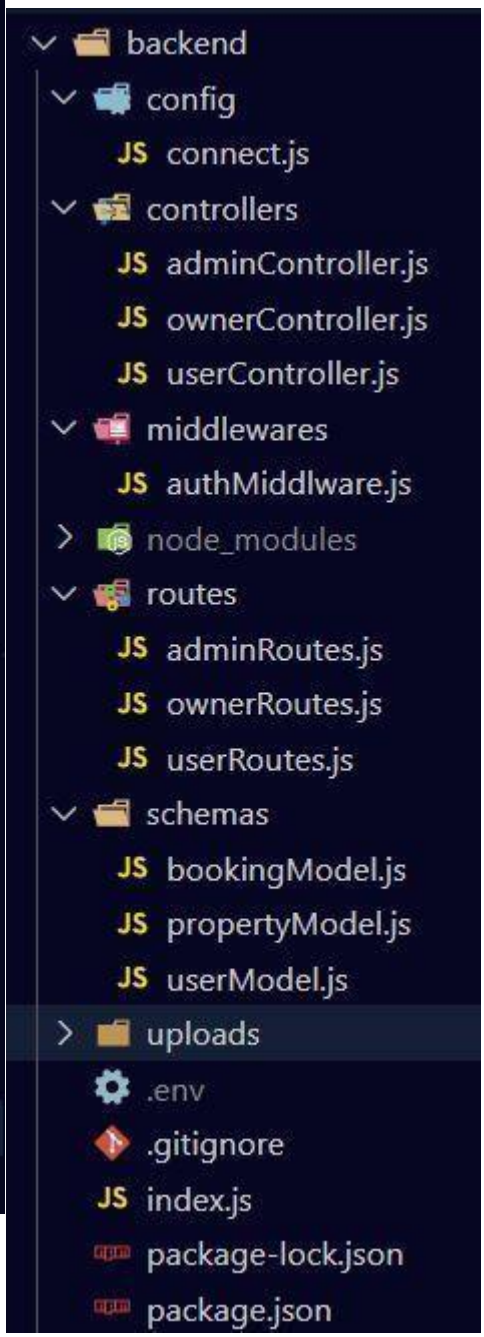
Owner:

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.

5. FOLDER STRUCTURE:


frontend
node_modules
public
index.html
src
images
p1.jpg
p2.jpg
p3.jpg
p4.jpg
modules
admin
AdminHome.jsx
AllBookings.jsx
AllProperty.jsx
AllUsers.jsx
common
ForgotPassword.jsx
Home.jsx
Login.jsx
Register.jsx
user
Owner
AddProperty.jsx
AllBookings.jsx
AllProperties.jsx
OwnerHome.jsx
renter
AllProperties.jsx


RenterHome.jsx
AllPropertiesCards.jsx
App.css
App.js
M
index.js
.gitignore
package-lock.json
package.json
README.md


backend
config
connect.js
controllers
adminController.js
ownerController.js
userController.js
middlewares
authMiddlware.js
node_modules
routes
adminRoutes.js
ownerRoutes.js
userRoutes.js
schemas
bookingModel.js
propertyModel.js
userModel.js
uploads
.env
.gitignore
index.js
package-lock.json
package.json

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

6. RUNNING THE APPLICATION:

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:**
  1. Create frontend and
  2. Backend folders

- ✔ **Installation of required tools:**

  1. Open the frontend folder to install necessary tools

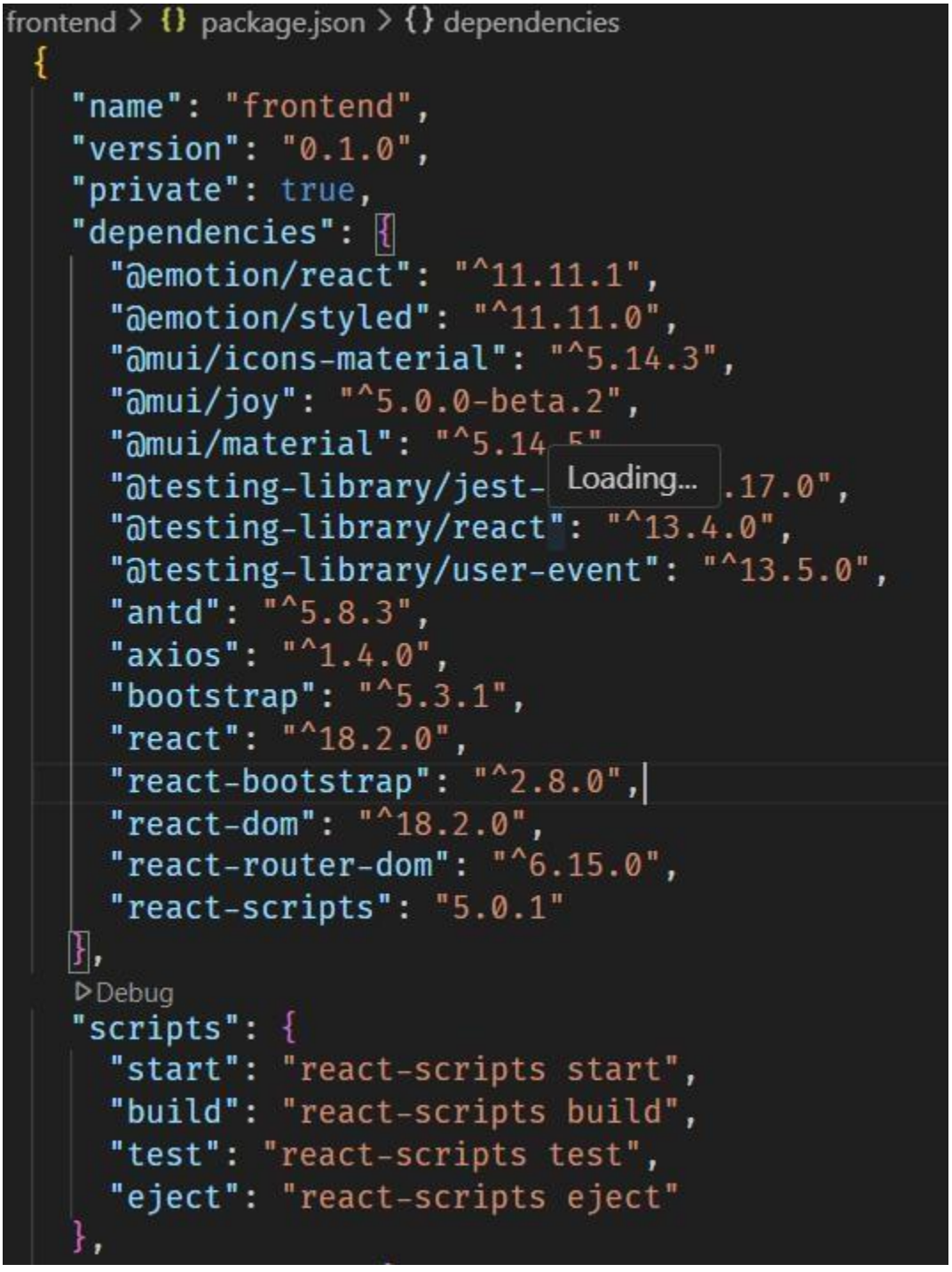
     For frontend, we use:
     - React
     - Bootstrap
     - Material UI
     - Axios
     - Moment
     - Antd
     - mdb-react-ui-kit
     - react-bootstrap

  2. Open the backend folder to install necessary tools

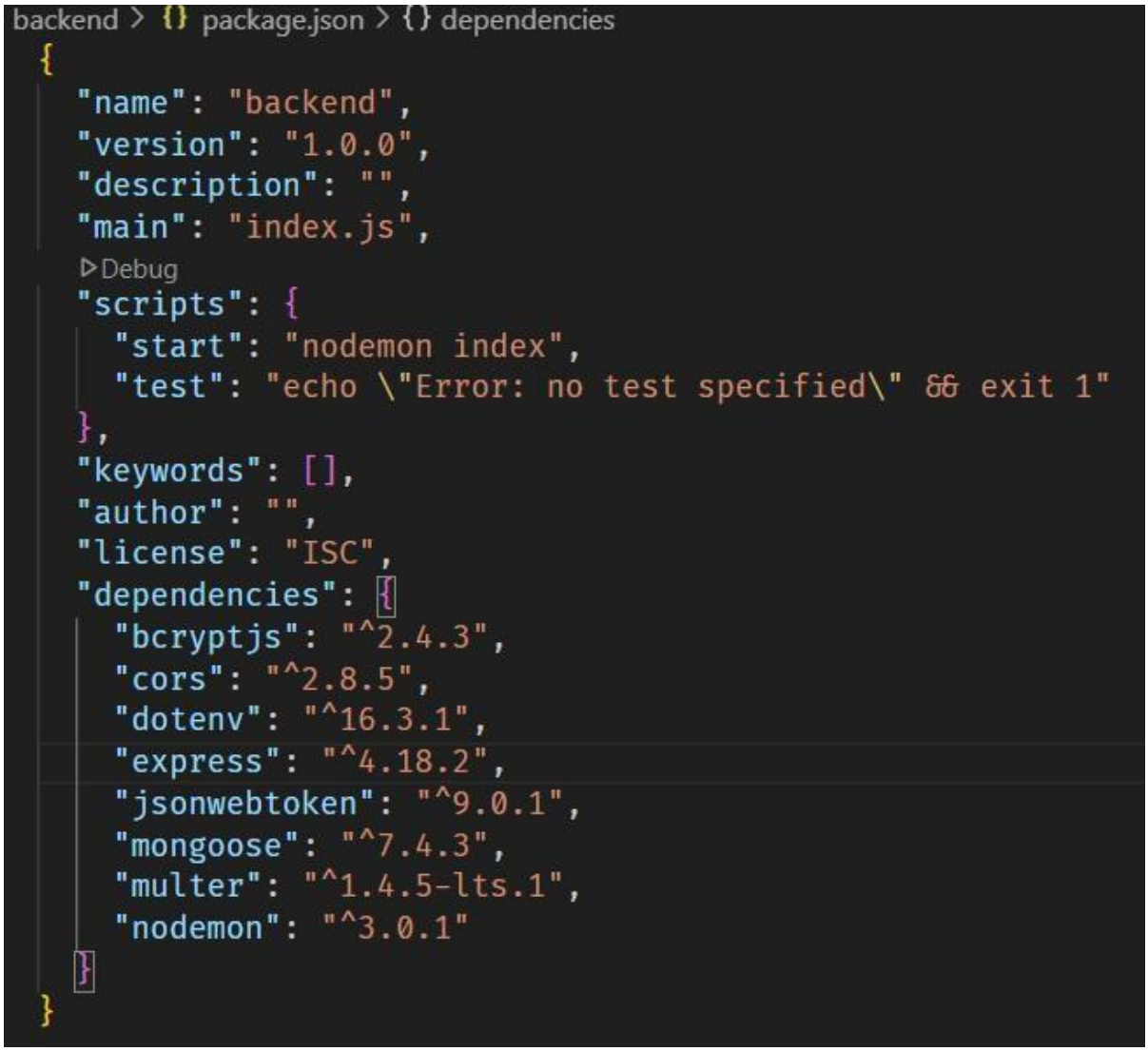
     For backend, we use:
     - cors
     - bcryptjs
     - express
     - dotenv
     - mongoose
     - Moment
     - Multer

- Nodemon
- jsonwebtoken

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



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

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

1. **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.

2. **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.

3. **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.

7. API DOCUMENTATION:

The backend of HouseHunt exposes several API endpoints categorized by user roles and features. While the exact endpoints aren't listed explicitly in the documentation, the functionality described allows us to infer the following:

Authentication APIs (All Users)

- POST /api/register – Register a new user (renter/owner).
- POST /api/login – Login and receive JWT token.
- GET /api/profile – Get current user profile using token.

Property Management (Owner Only)

- POST /api/properties/add – Add a new property.
- GET /api/properties/owner – Get properties added by the logged-in owner.
- PUT /api/properties/update/:id – Update a property.
- DELETE /api/properties/delete/:id – Delete a property.
- PUT /api/properties/status/:id – Change availability/status.

Property Browsing (Renter Only)

- GET /api/properties – View all available properties.
- GET /api/properties/:id – View specific property info.
- POST /api/properties/book/:id – Send booking request.

Booking Management

- GET /api/bookings/renter – View bookings made by renter.
- GET /api/bookings/owner – View all booking requests for an owner's properties.
- PUT /api/bookings/update/:id – Owner can approve/reject a booking.

Admin APIs

- GET /api/admin/users – View all users.
- PUT /api/admin/approve-owner/:id – Approve a user as an owner.
- GET /api/admin/bookings – Monitor all platform activity/bookings.

All routes are secured with JWT middleware, with role-based access control applied

8. AUTHENTICATION

The House Hunt platform uses **JWT (JSON Web Token)** for secure user authentication:

Login Flow:

1. User logs in with email & password.
2. Server validates credentials and sends back a **JWT token**.
3. Frontend stores the token (typically in local storage).
4. Token is sent in headers for all protected API calls.

Authorization Types:

- **Renter:** Can view properties, book houses, and see booking status.
- **Owner:** Can add/update/delete properties and approve bookings.
- **Admin:** Has full control to manage user roles and monitor app activity.

Middleware:

- authMiddleware.js ensures that routes are accessed only with valid tokens.
- Role-based checks ensure only the correct user type accesses specific routes.

9. USER INTERFACE

Technologies Used:

- **React.js**
- **Material UI**
- **Bootstrap**
- **Ant Design**
- **React-Bootstrap**
- **Axios (for API calls)**

Main Pages:

1. **Landing Page:** Welcome message and login/register options.
2. **Login/Register Page:** Forms for user authentication.
3. **Dashboard (role-based):**
    - **Renter View:** Browse all properties, book houses, view booking history.
    - **Owner View:** Manage properties (CRUD), check bookings.
    - **Admin View:** Approve owners, monitor all users/bookings.
4. **Property Detail Page:** High-quality images, 360° tour (if available), owner

contact, booking form.

5. **Booking History Page:** Status: Pending, Approved, Rejected.

10. TESTING

The original document does not detail testing, so here are suggestions based on common MERN practices:

Manual Testing:

- **Frontend:** Via browser – test property listings, booking forms, dashboard functionalities.
- **Backend:** Use **Postman** to manually test each API.

Automated Testing (Recommended Additions):

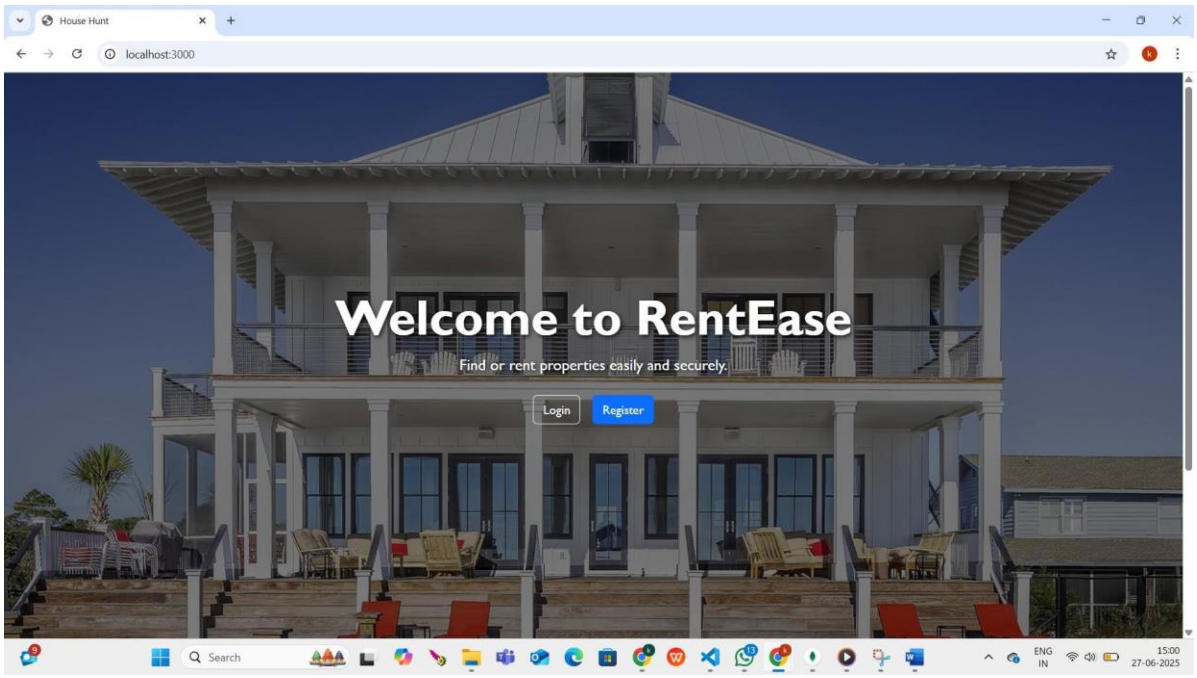
- **Backend:**
  - Use **Jest** and **Supertest** for API testing.
- **Frontend:**
  - Use **React Testing Library** and **Jest** to test UI components and API integration.

11. SCREENSHOTS AND DEMO:

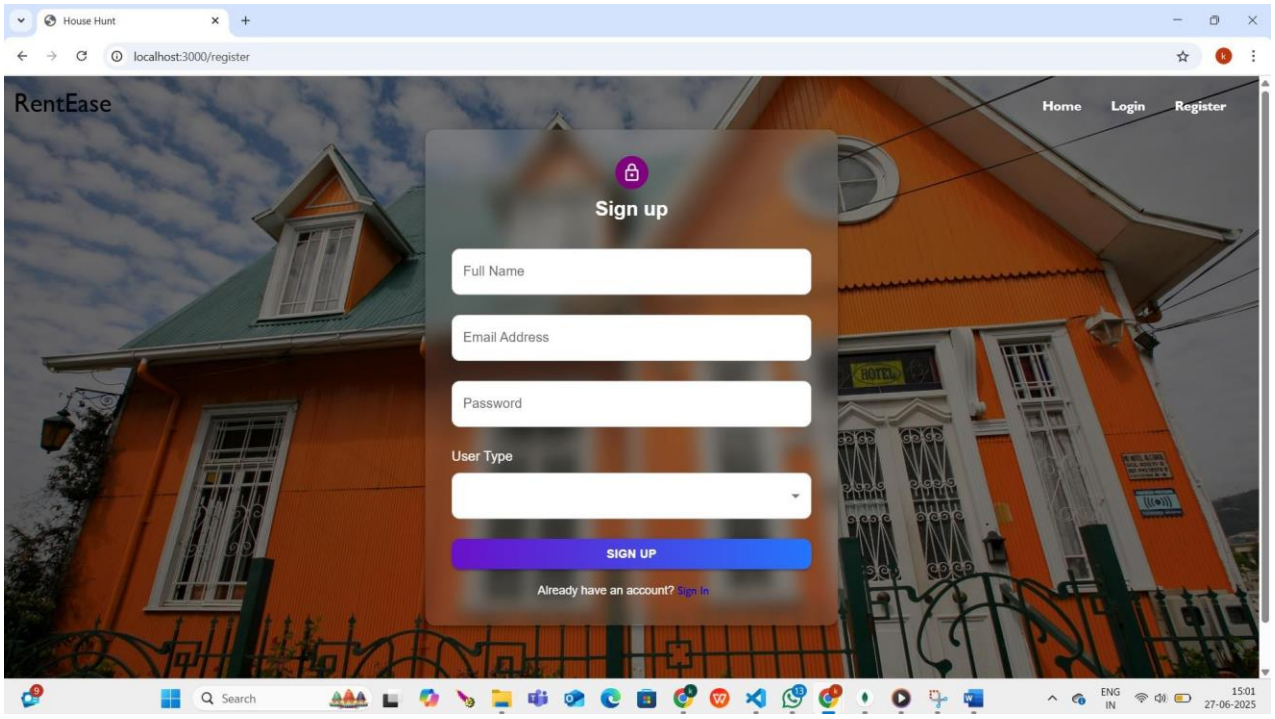
DEMO LINK:

https://drive.google.com/file/d/1PqJBucNH4oAnXBrWI3Rk4_Dw7iDupMlg/view?usp=sharing

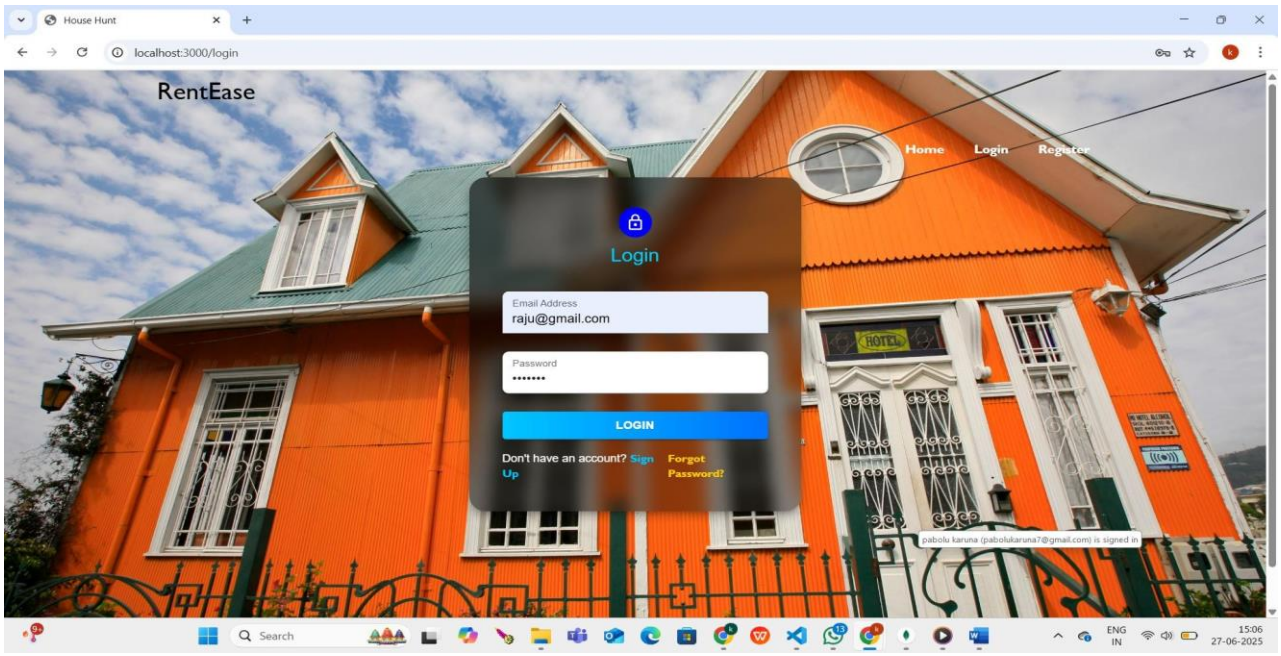
HOME PAGE:



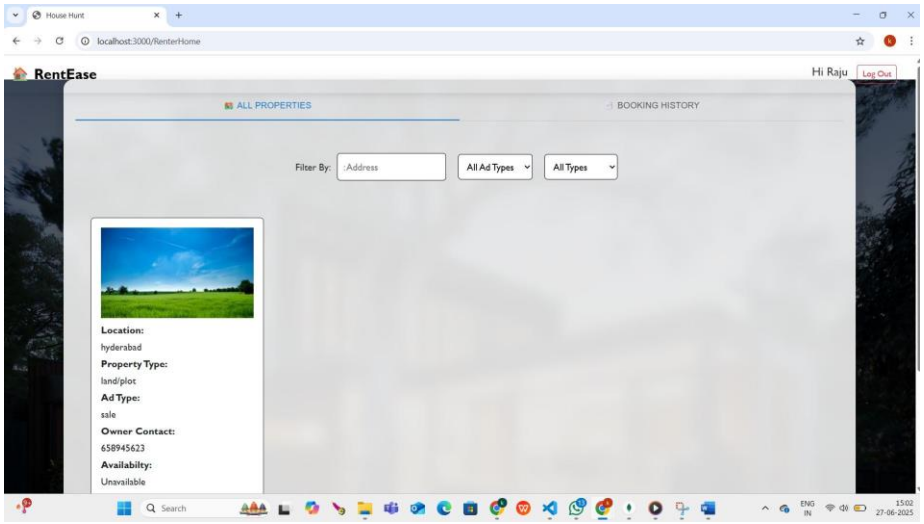
REGISTRATIONPAGE

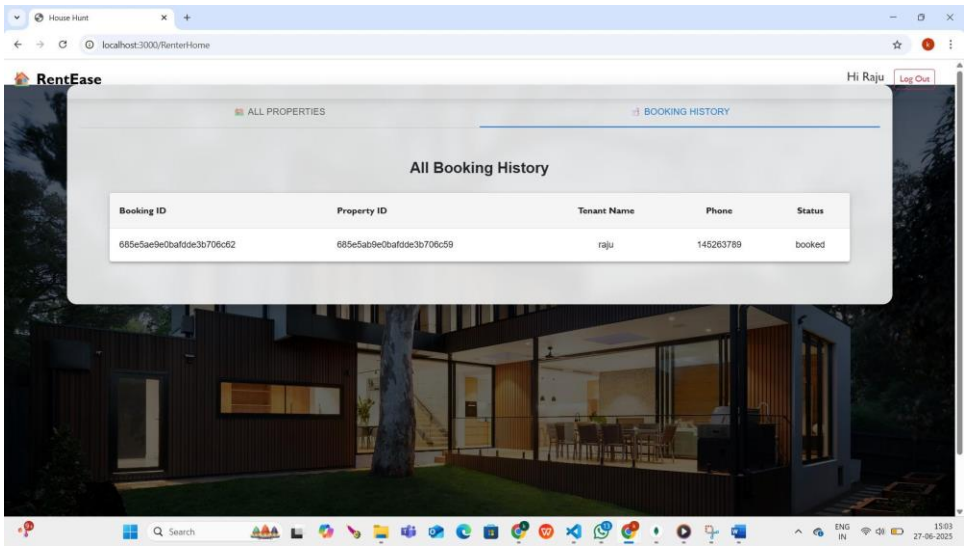


LOGINPAGE

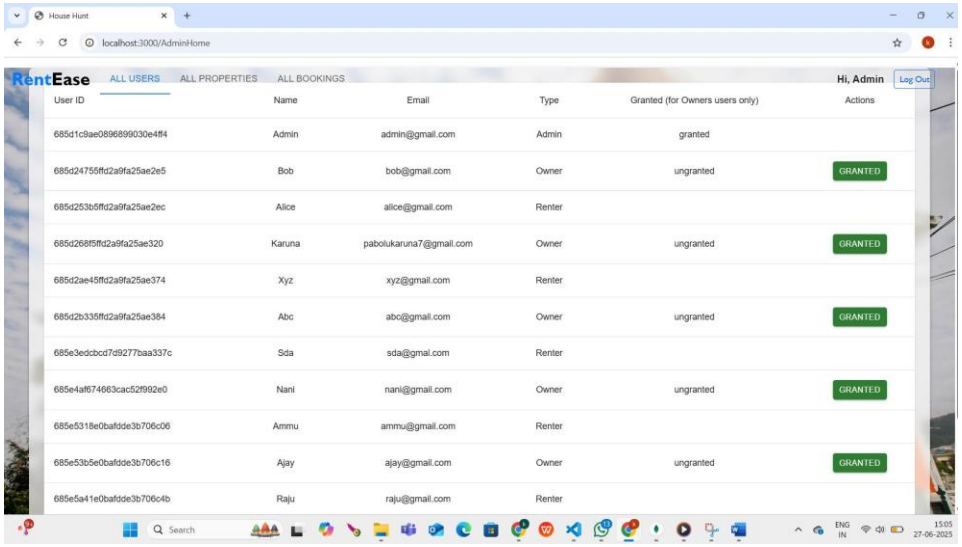


RENTER HOME PAGE

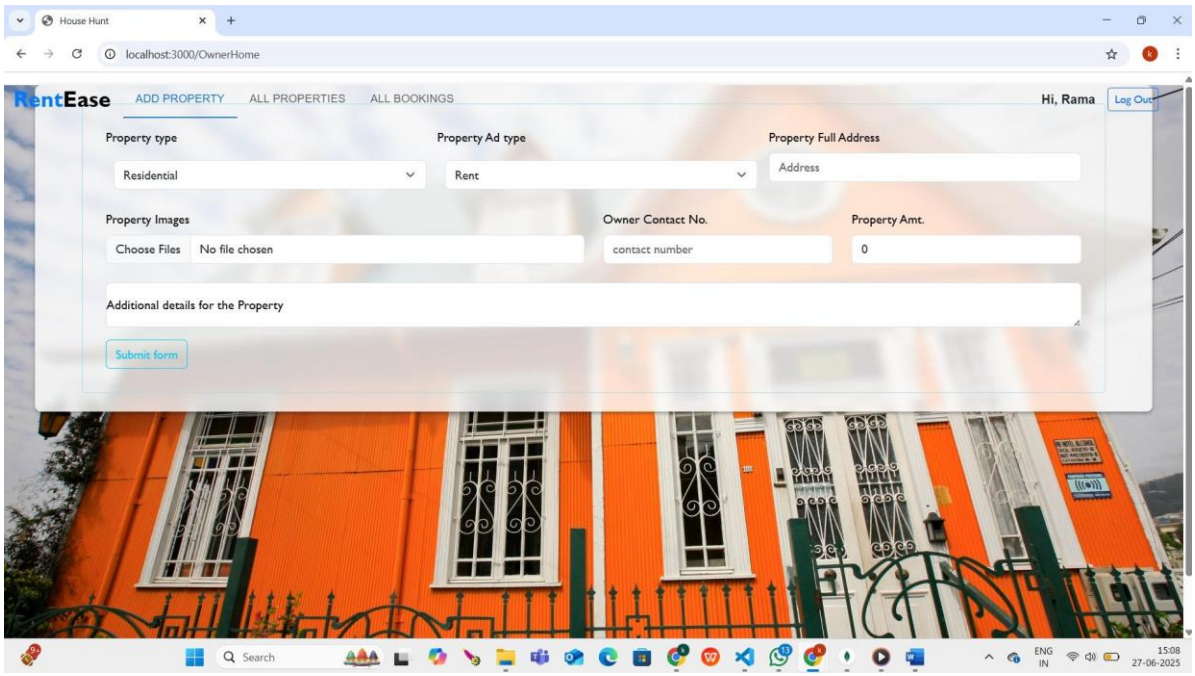


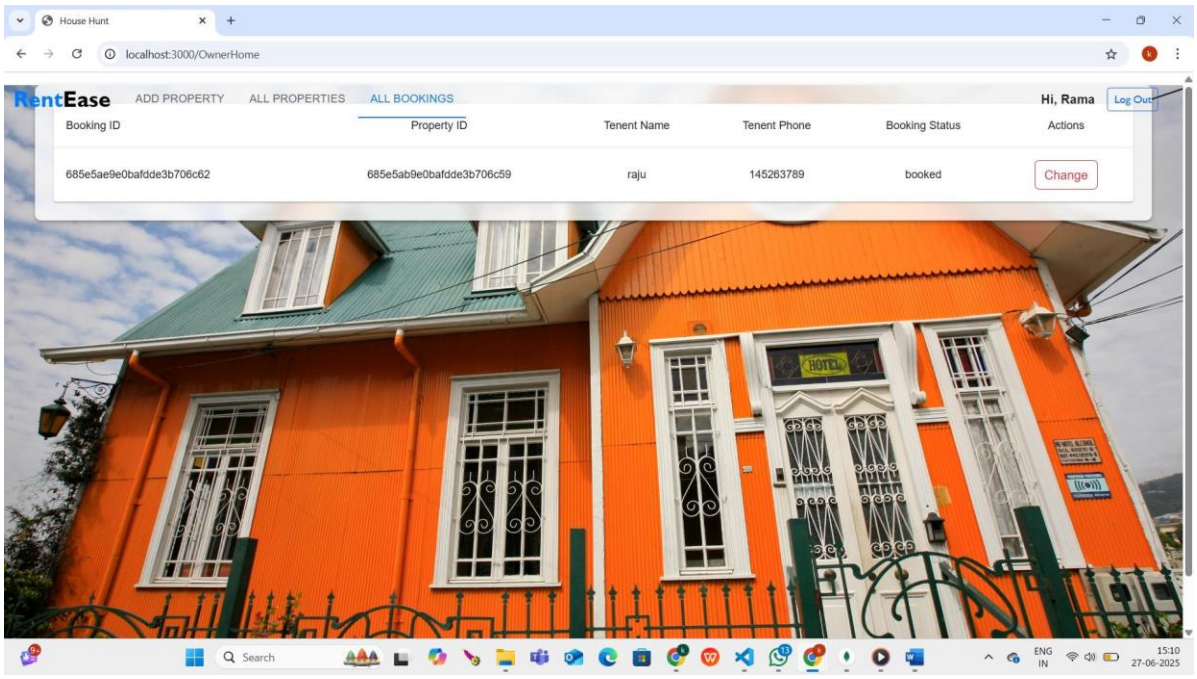


ADMIN HOME PAGE:



OWNER HOME PAGE:





12. KNOWN ISSUES

- JWT token expiry handling not mentioned.
- No detailed error handling or toast/alert system described for UI feedback.
- Admin flow seems mixed with legacy references (mentions "doctors" – possibly from a reused template).
- Booking approval flow may not include email notifications.

13. FUTURE ENHANCEMENTS

Suggestions for future improvement:

1. **Payment Gateway Integration:** Allow users to make payments directly for bookings via Razorpay/Stripe.
2. **Chat Feature:** Enable live chat between renter and owner using socket.io or third-party chat APIs.
3. **Ratings and Reviews:** Users can rate properties and owners based on experience.
4. **Mobile App Version:** Develop the same in React Native for better accessibility.
5. **Property Map View:** Use Google Maps API to display properties visually.
6. **Admin Analytics Dashboard:** Add graphs and stats on user activity, bookings, and top locations.