Full Stack Development with MERN

1. INTRODUCTION:

The House Hunt Project is a comprehensive platform designed to simplify and streamline the process of searching for residential properties. In today's fast-paced world, finding the right home that matches a buyer's preferences, budget, and location requirements can be challenging. A house rent app is typically a mobile or web application designed to help users find rental properties, apartments, or houses for rent.

❖ PROJECT TITLE:

House Hunt: Finding Your Perfect Rental Home

❖ TEAM MEMBERS:

Team Leader: velakaturi Pavithra- Frontend development

Team member: Nelavayi Kavya Sree- Backend development Team

member: Kummarakunta Muneswari- Testing & deployment Team

member: Malkogi Navya Sree- Final documentation

2. PROJECT OVERVIEW:

House Hunt is a full-stack web application designed to streamline the process of renting and listing residential properties. It serves as a digital bridge between property owners and potential tenants, offering a user-friendly interface and robust backend functionality.

The project also supports property owners and real estate agents in listing, managing, and promoting properties.

❖ PURPOSE:

- The primary purpose of the House Hunt Project is to streamline and modernize the process of finding, listing, and managing residential properties by creating a centralized, digital platform that connects property seekers with sellers and agents.

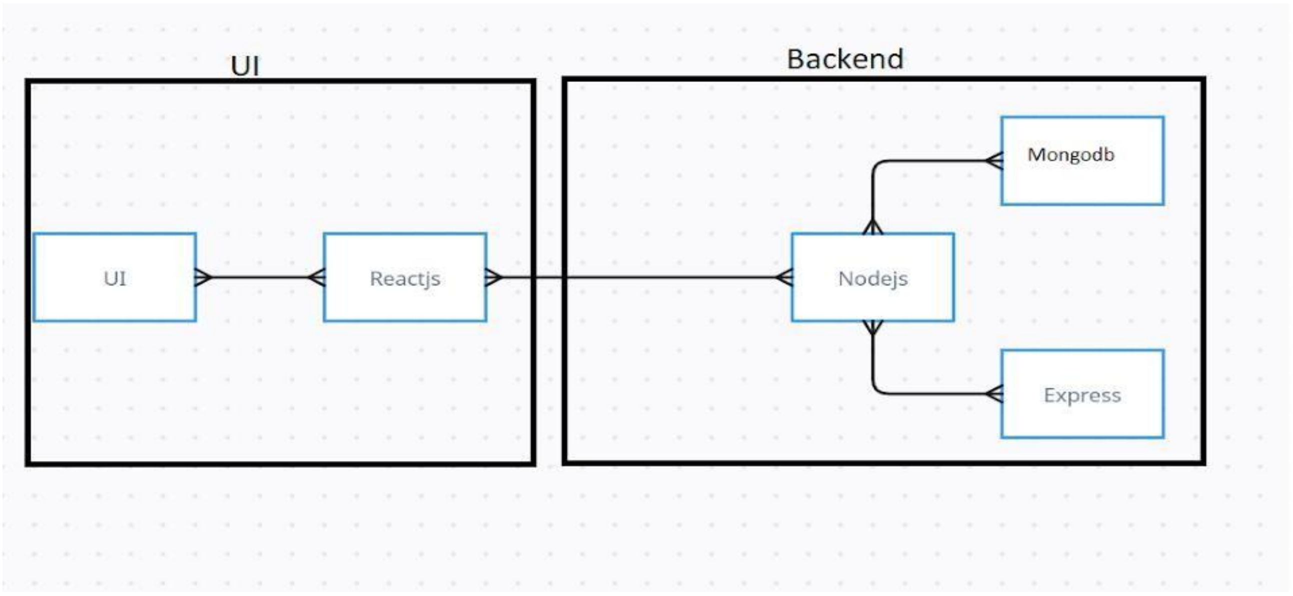
1. Empowering Users with Information: House Hunt provides detailed listings, market insights, and property data, helping users make informed decisions about buying, selling, or renting homes.

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.

FEATURES:

- **User Authentication**: Secure login and registration for both tenants and property owners
- **Property Listings**: Owners can add, edit, and delete property details including images, rent, amenities, and location
- **Search & Filters**: Users can search by city, price range, number of rooms, and property type
- **Google Maps Integration**: Visualize property locations on an interactive map
- **Responsive Design**: Optimized for mobile and desktop devices
- **Admin Dashboard**: Admins can manage users, listings, and reported content
- **Image & Video Upload**: Showcase properties with rich media content
- **Booking System**: Tenants can schedule property visits or send inquiries directly to owners

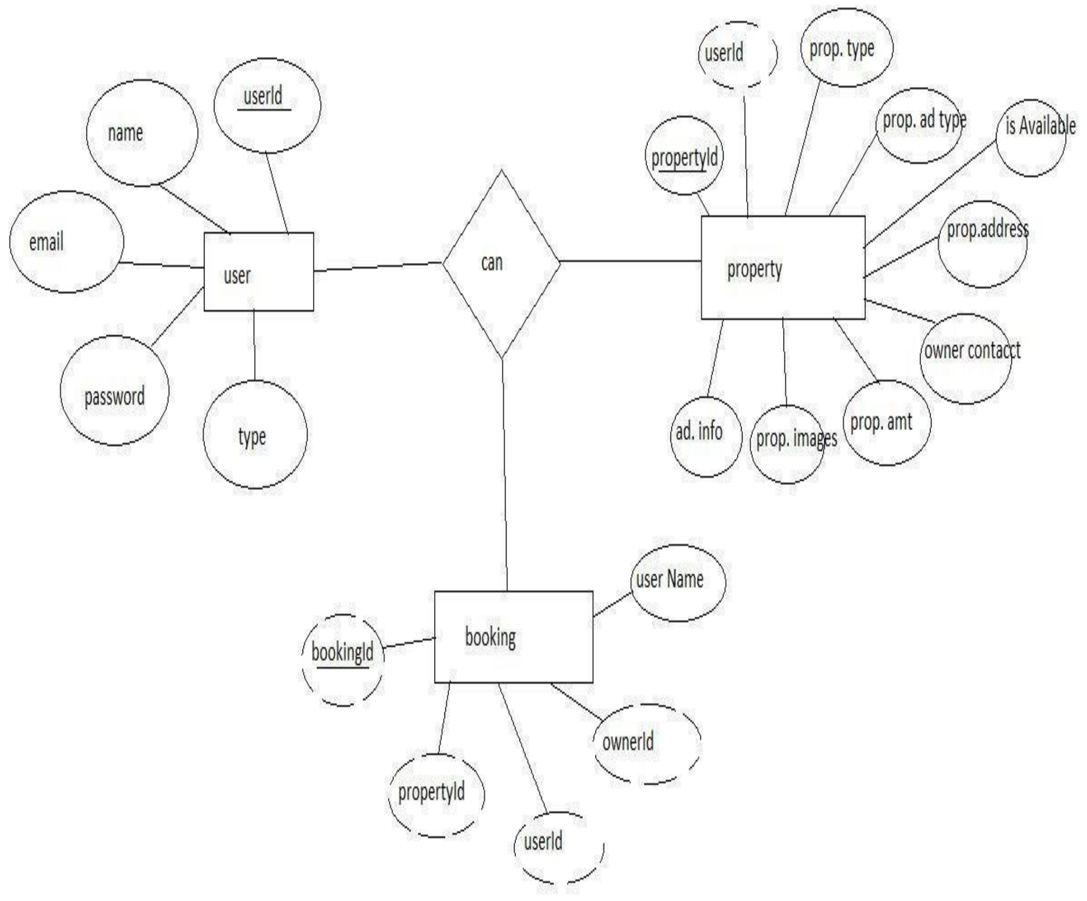
3. ARCHITECTURE:



✓ 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 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 and handle the server-side logic and Communication.
- ✓ The frontend and backend components, along with moment, Express.js, and MongoDB, form a comprehensive technical architecture for our House. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring, smooth and also immersive booking an appointment and many more experience for all users.

ER DIAGRAM:



4. SETUP INSTRUCTIONS:

The House Hunt Project is a full-stack web application developed to simplify the process of property searching and listing. This section outlines the step-by-step instructions to set up the project in a local development environment.

- **Frontend**: Deploy using Netlify, Vercel, or Firebase Hosting.
- **Backend**: Deploy using Heroku, Render, or Railway.
- **Database**: Use MongoDB Atlas for cloud database access.
- Before setting up the project, ensure the following tools are installed on your system:

    - **Node.js** and **npm** – for running JavaScript on the backend and managing packages
    - **MongoDB** – used as the database (either locally or via MongoDB Atlas)
    - **Git** – for cloning the project repository
    - **Code Editor** – such as Visual Studio Code

PRE-REQUISTIC:

- 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:

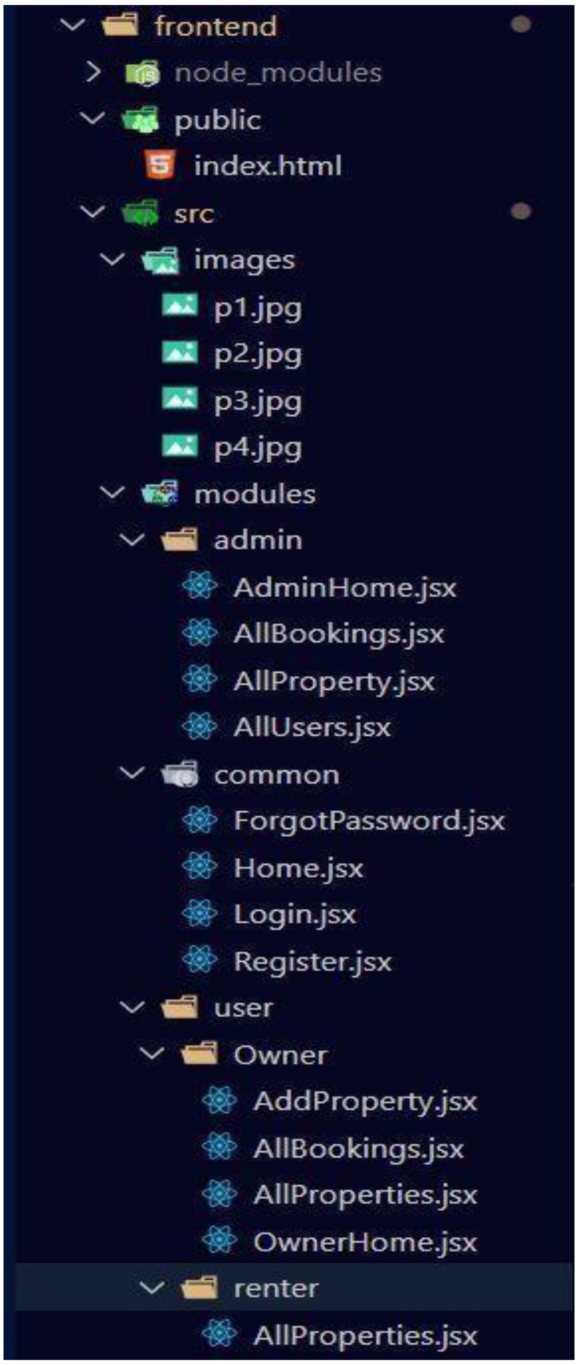
Utilize React.js to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard.

- For making better UI we have also used some libraries like material UI and Boostrap

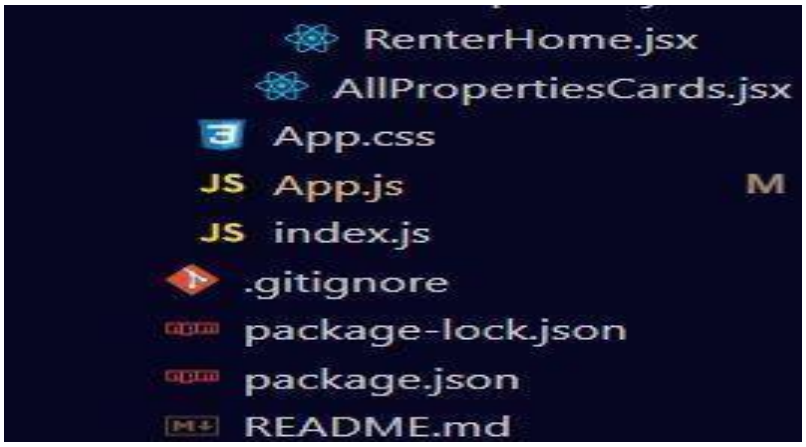
5. FOLDER STRUCTURE:

The **House Hunt Project** is organized as a full-stack web application, consisting of two main parts: the **frontend** and the **backend**. The frontend is built using **React.js** and is located in the client directory. Within this, the public folder contains static files such as index.html and favicon icons, while the src folder includes all the source code. This includes reusable UI components (components), complete page views (pages), API communication services (services), context providers for global state management (context), and helper functions in a utils directory. The entry point of the application is index.js, and routing is handled in App.js.

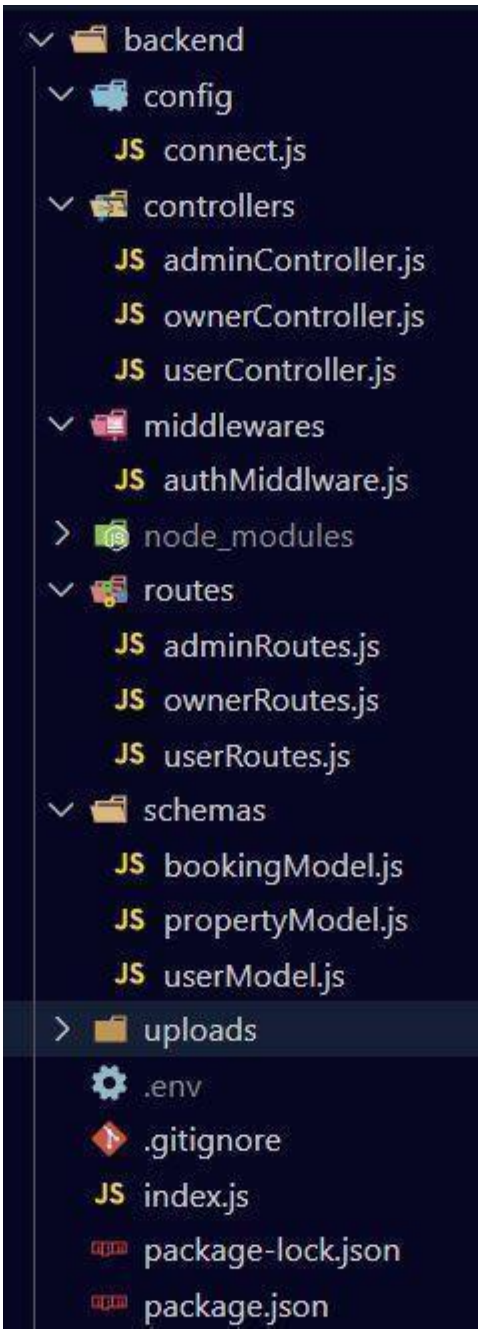
The backend, built with **Node.js and Express.js**, is found in the server directory. It contains various folders such as routes (defining API endpoints like user authentication and property listing), controllers (containing the logic for handling route requests), and models (which define MongoDB schemas using Mongoose). The middleware folder includes custom middleware for tasks like JWT authentication and error handling.



```
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
```

6. RUNNING THE APPLICATION:

- **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:**

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

User Authentication:

- Implement user authentication using JWT or session-based methods.
- Create routes and middleware for user registration, login, and logout.
- Use authentication middleware to protect routes requiring user authorization

- **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

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.

- **Installation of required tools:**

1. **React:**
   The UI is broken into reusable components like Navbar, Property Card, Search Bar, and Listing Form, making the code modular and easier to maintain. React use State and use Effect hooks manage dynamic data like search filters, user input, and API responses. React Router handles navigation between pages like Home, Login, Register etc

2. **Bootstrap:**
   Bootstrap's 12-column grid ensures that property listings, search bars, and forms adapt seamlessly to different screen sizes—from desktops to smartphones.
   Elements like **modals**, **cards**, **carousels**, and **navbars** are used to display property details, image galleries, and navigation menus with minimal custom CSS.

3. **Material UI:**

   In the House unt project**,** Material UI (MUI) is used to give the frontend a sleek, modern look while speeding up development with ready-made, customizable components.

4. **Axios:**

Axios is used to handle HTTP requests between the React frontend and the Node.js/Express backend. It simplifies:

- Fetching property listings from the server
- Submitting login/signup forms
- Posting new property data
- Handling errors and loading states gracefully

5. **Moment:**
   Moment is typically used for date formatting and manipulation. In House Hunt, it might be used to:

- Format listing dates
- Display appointment times in a readable format

6. **Antd:**
   Antd provides a rich set of enterprise-grade UI components. In House Hunt, it can be used for:

- Elegant forms for login, registration, and property submission
- Tables for admin dashboards
- Modals, notifications, and date pickers for user interactions

7. **mdb-react-ui-kit:**
   This is a Material Design Bootstrap UI kit for React. It offers:

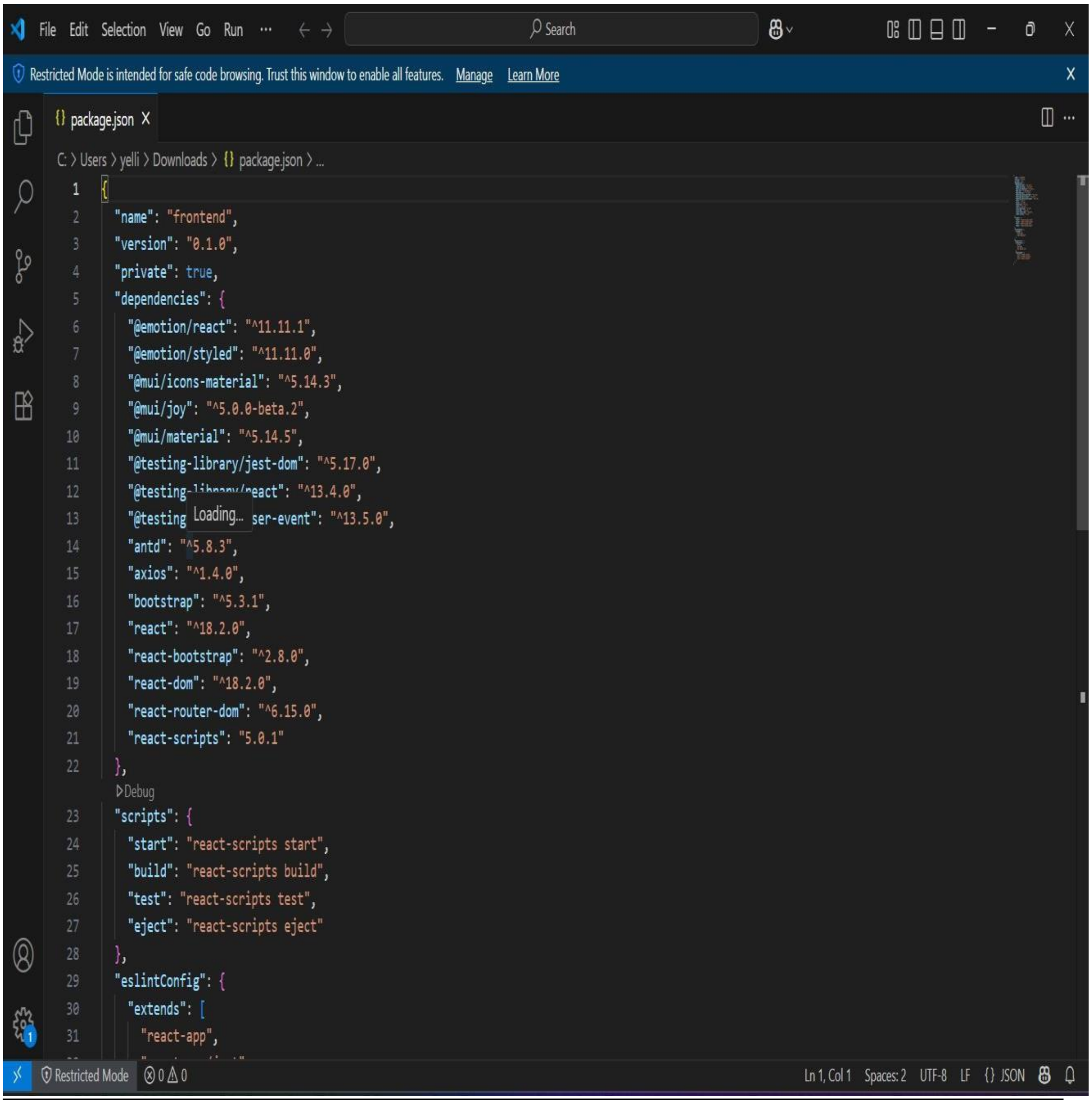
- Pre-styled components like cards, buttons, spinners, and footers
- Aesthetic enhancements for property listings and user dashboards
- Easy integration with Bootstrap 5 and React 18

8. **react-bootstrap:**
   React-Bootstrap brings Bootstrap's styling into React component model. It's often used for:

- Navigation bars, grids, and responsive layouts
- Alerts, modals, and form controls
- Ensuring mobile-first design without writing custom CSS

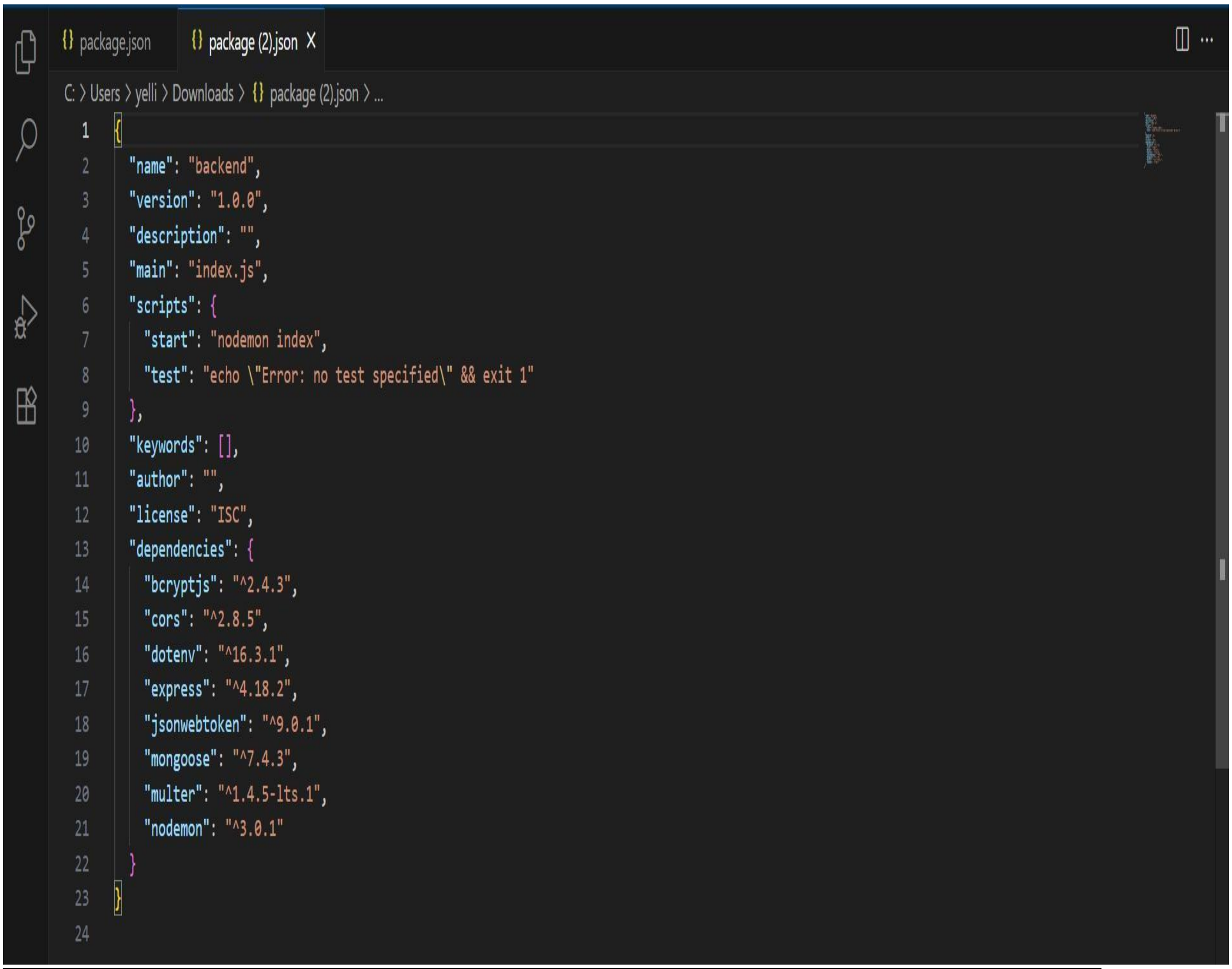
7. API DOCUMENTATION:

Authentication APIs


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
package.json
C: > Users > yelli > Downloads > {} package.json > ...
Restricted Mode ⊗ 0 ⚠ 0
Ln 1, Col 1 Spaces: 2 UTF-8 LF {} JSON

```
{
  "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 Loading... ser-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"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
```

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


{} package.json   {} package (2).json

C: > Users > yelli > Downloads > {} package (2).json > ...

```json
{
  "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"
  }
}
```

The House Hunt project is a full-stack web application designed to simplify the process of listing and searching for rental properties. Here's a quick overview of its backend architecture:

- Backend Framework: Built using Node.js with the Express.js framework, which handles routing, middleware, and server-side logic.
- Database: Uses MongoDB, a NoSQL database, to store property listings, user profiles, and other dynamic data.
- API Integration: RESTful APIs are implemented to manage CRUD operations for listings, user authentication, and image uploads**.**
- Authentication: Includes secure user login and registration features, likely using JWT (JSON Web Tokens) or sessions for managing user sessions.
- File Handling: Supports image uploads for property listings, possibly using middleware

- Communication: Uses Axios for making HTTP requests between the frontend and backend.

7. AUTHENTICATION:

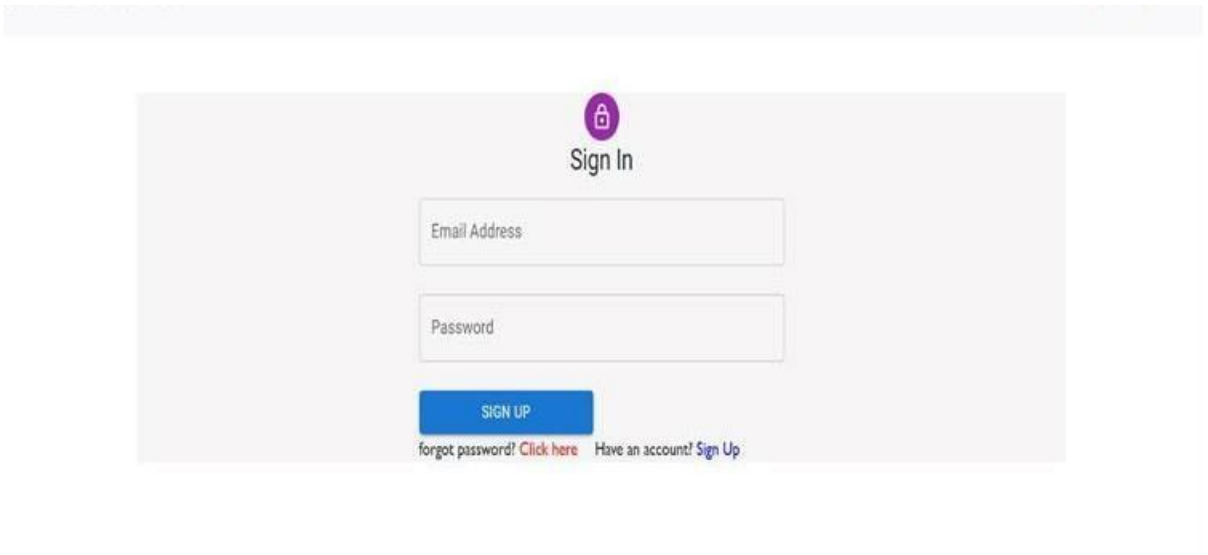
Authentication in the House Hunt project ensures that only registered users can access certain features like posting or saving property listings. Here's how it's typically structured:

- Registration & Login: Users can sign up with their email and password. During login, credentials are verified against the database.
- Password Security: Passwords are hashed using libraries like bcrypt before being stored in MongoDB, ensuring they're never saved in plain text.
- JWT (JSON Web Tokens): After successful login, a JWT is generated and sent User to the client. This token is used to authenticate future requests without needing to log in again.
- Protected Routes: Backend routes (like creating or editing listings) are protected using middleware that checks for a valid JWT.
- Session Management: Tokens may have expiration times, and refresh tokens can be used to maintain sessions securely.

9. USER INTERFACE:

• Providing screenshots or GIFs showcasing different UI features.

Authentication:



10. TESTING:

Testing in the House Hunt project ensures that all backend and frontend components work reliably and securely. Here's a structured overview of how testing might be implemented

Types of Testing

- **Unit Testing**: Focuses on individual functions or modules (e.g., user authentication, property listing APIs). Tools like Jest or Mocha are commonly used.
- **Integration Testing**: Verifies that different modules (like the database and API routes) work together as expected.
- **End-to-End (E2E) Testing**: Simulates real user scenarios—like signing up, logging in, and posting a property—using tools like Cypress or Selenium.
- API Testing: Ensures RESTful endpoints return correct responses and handle errors gracefully. Tools like Postman or Super test are often used.
- **Security Testing**: Checks for vulnerabilities like SQL injection, broken authentication, or insecure data storage.
- **Performance Testing**: Evaluates how the app handles load, especially User **Authentication**: Validates login, signup, and token verification.

Installation of required tools:

1. Open the frontend folder to install necessary tools

   For frontend, we use:

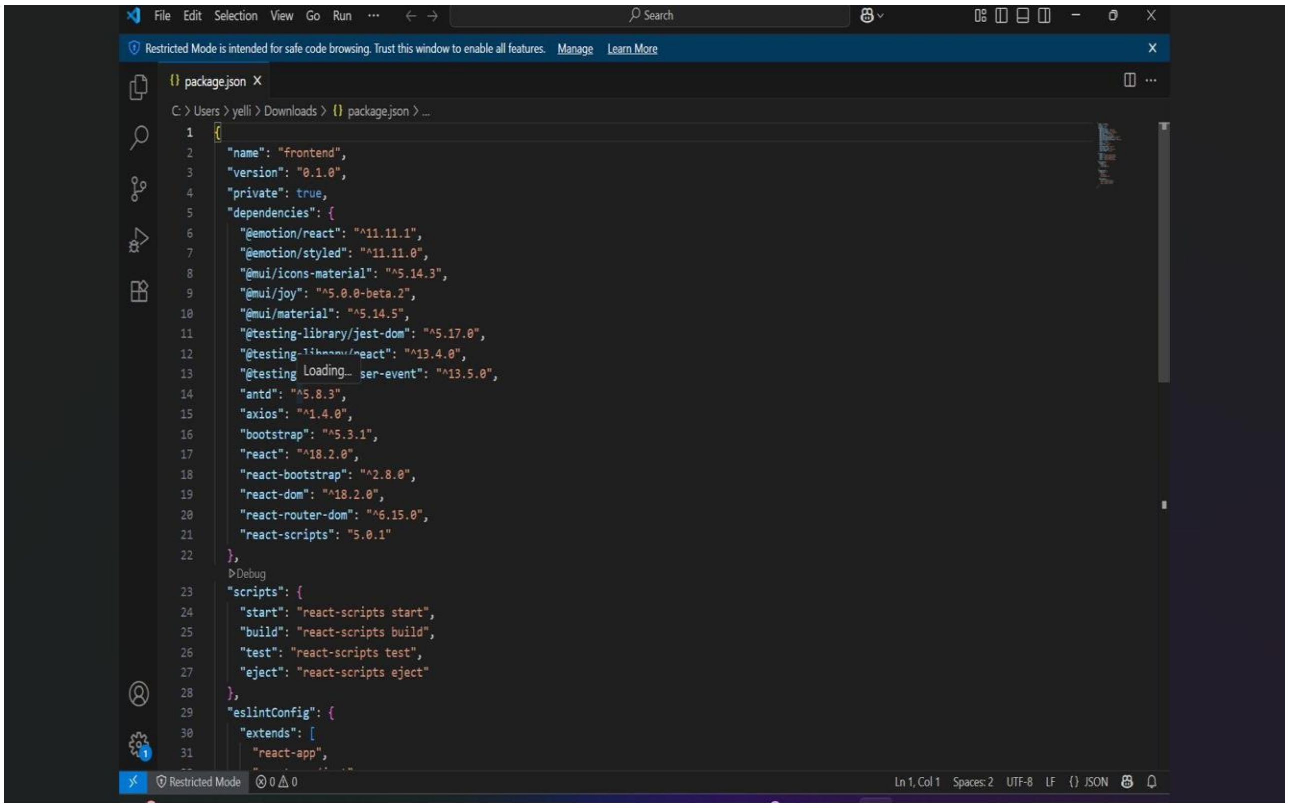
   - React
   - Bootstrap
   - Material UI
   - Axios
   - react-bootstrap

2. Open the backend folder to install necessary tools

   For backend, we use:

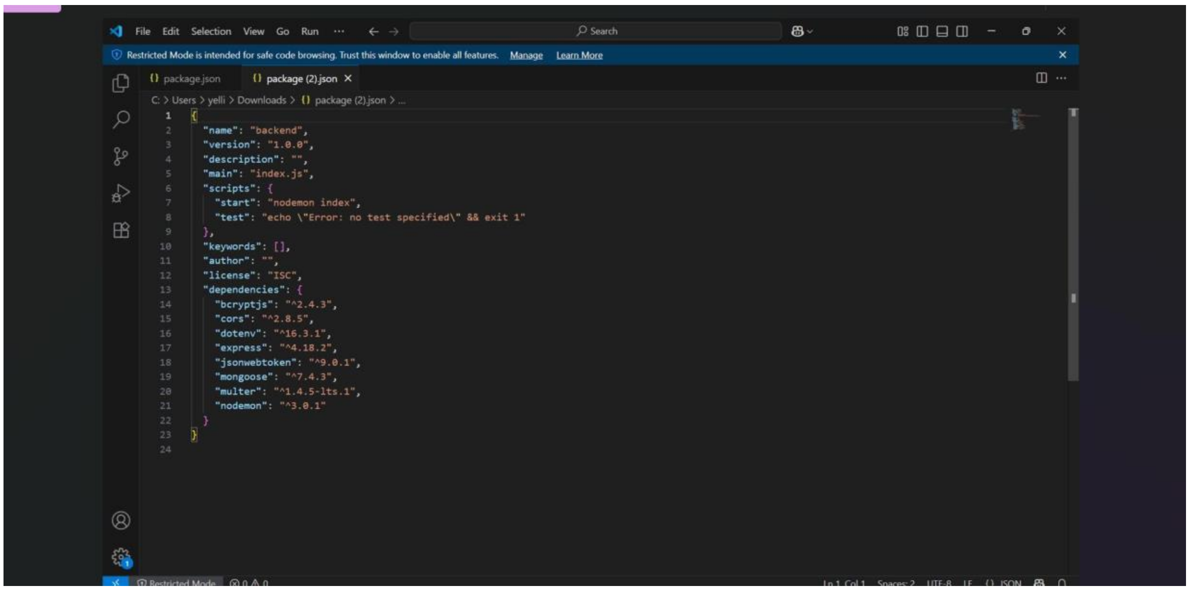
   - Express Js
   - Node JS
   - MongoDB
   - Mongoose
   - Cors
   - Bcrypt

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



```
{
  "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 Loading... ser-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"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
```

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



```
{
  "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"
  }
}
```

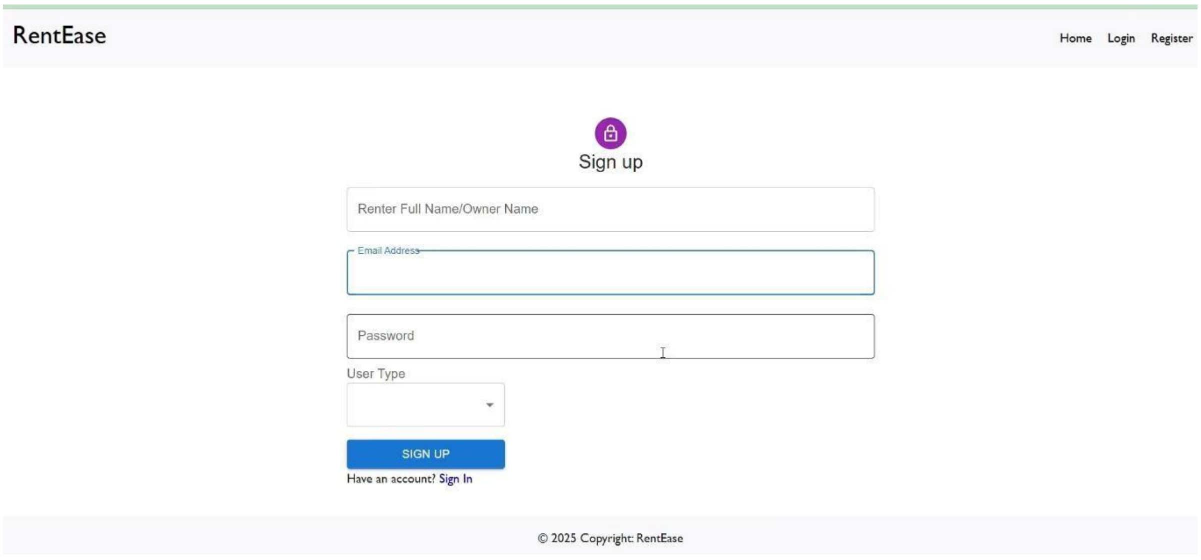
11. SCREENSHOTS OR DEMO:

- Providing screenshots or a link to a demo to showcase the application.

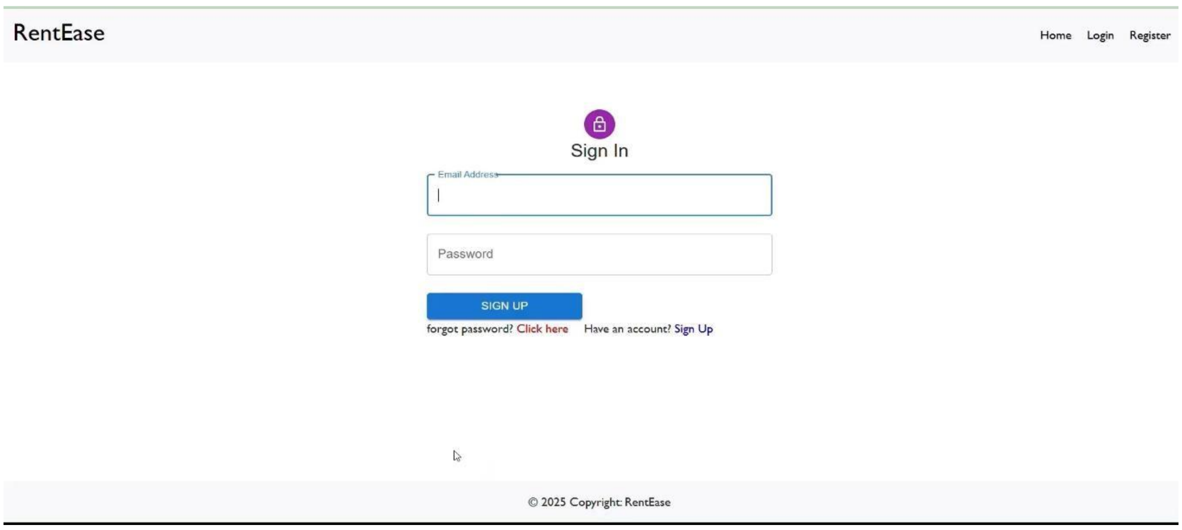
PROJECT IMPLEMENTATION

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 images provided below.

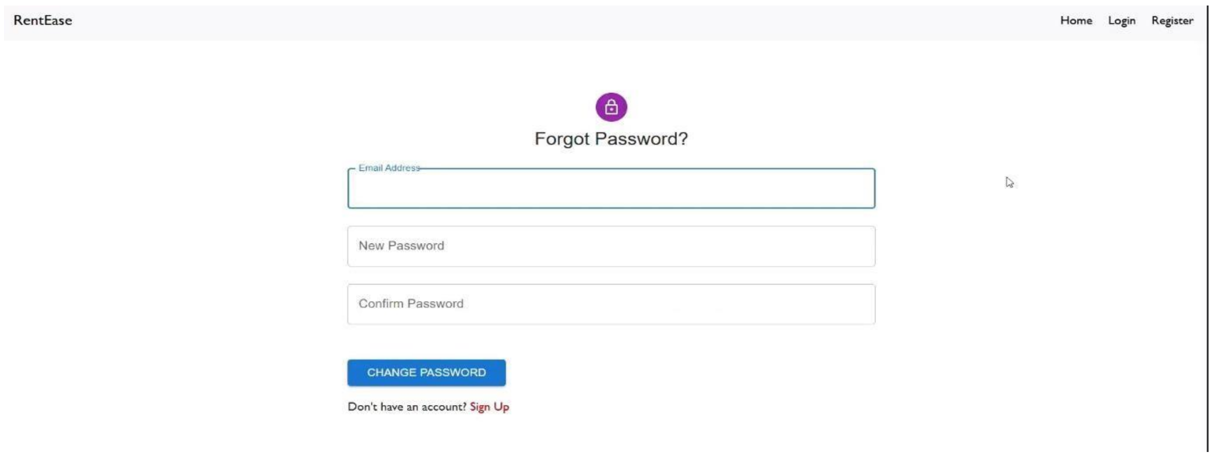
Register or sign up:



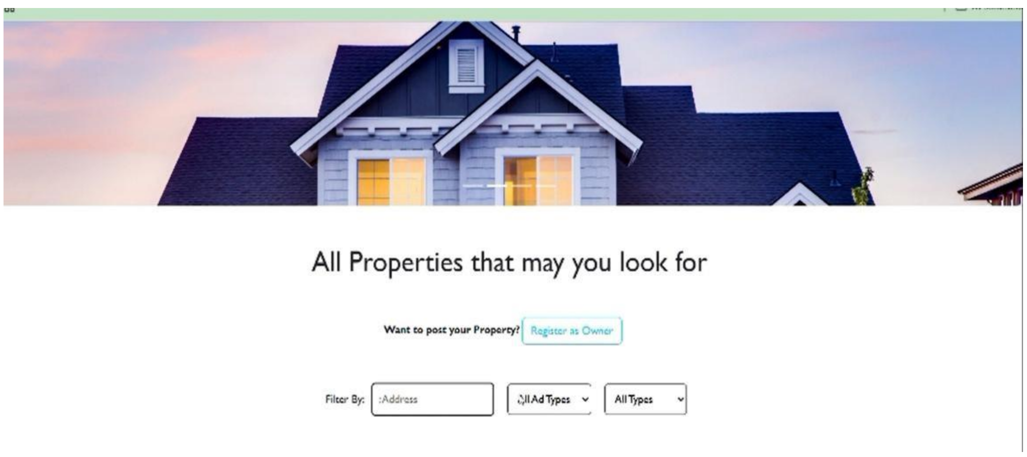
Login:



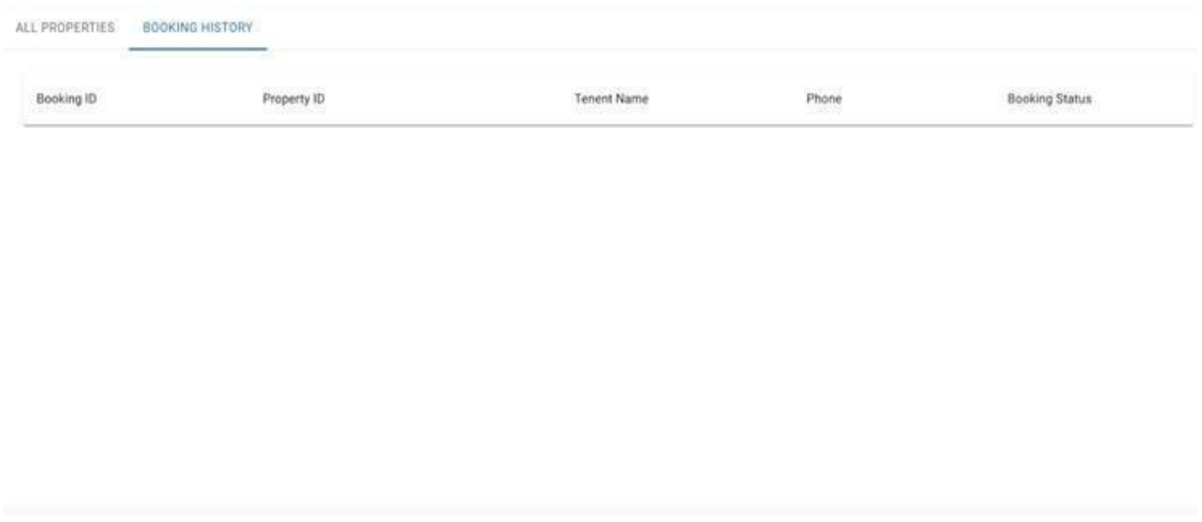
Password change:



Properties:


All Properties that may you look for
Want to post your Property? Register as Owner
Filter By: :Address
All Ad Types
All Types

Booking History:



Booking ID	Property ID	Tenent Name	Phone	Booking Status

No Properties available at the moment.

12. KNOWN ISSUES:

- **Geolocation Integration Challenges**: Some developers reported difficulties integrating geolocation services like Radar.io and Google Maps API, especially when converting tenant addresses into coordinates and displaying them accurately on the map.
- **Network Instability:** During development or deployment, unstable network conditions caused delays or failures in API calls and data fetching.
- **Image Upload Handling:** Managing image uploads securely and efficiently—especially with large files or slow connections—can be tricky without proper middleware and validation.
- **Authentication Bugs:** Issues may arise with token expiration, refresh logic, or inconsistent session handling if JWTs aren’t implemented carefully.
- **Database Schema Migrations:** In some versions, rolling back database changes was problematic due to the lack of reversible migrations, especially when using append-only migration strategies.
- **UI Responsiveness:** While the app is designed to be responsive, some users noted layout glitches on smaller screens or older browsers.

13. FUTURE ENHANCEMENTS:

- **AI-Powered Recommendations:**
  Suggest listings based on user behavior, preferences, and search history.
- **Price Prediction Engine:** Use machine learning to estimate fair rental prices based on location, amenities, and market trends.
- **Image Recognition:**

Automatically tag and categorize property images using computer vision.

- **Verified Listings**:
  Add a verification badge for listings reviewed by admins or verified landlords.
- **Tenant Background Checks**:
  Integrate third-party services to offer optional background screening.
- **Map-Based Search**: Integrate interactive maps (e.g., Google Maps or Leaflet.js) for visual property exploration.
- **Neighbourhood Insights**: Show nearby amenities, schools, crime rates, and commute times.
- **Real-Time Chat**: Enable messaging between tenants and property owners using Web sockets or services like Firebase.
- **Saved Searches & Alerts**: Let users save search criteria and receive notifications when matching properties are listed