

1. INTRODUCTION

Project Title: HouseHunt – Find Your perfect space-Rent , Book and Live withEase

Team Members:

1. **Team Leader: Pothaboina Bhuvana** – Project coordination,frontend/backend integration,authentication,bookig system
2. **Team Member: Pirati Harsha Sri** – User registration/login , form validations, homepage responsiveness.
3. **Team Member: Pichuka Varshitha**– Owner dashboard functionality , proerty edit / delete,image upload logic
4. **Team Member:Peteti Chandana Priya Sai Durga**– Admin approval system , backend schema design , manual testing & bug fixing

2. PROJECT OVERVIEW

HouseHunt is a role-based web platform designed to simplify the house rental process. It bridges the gap between renters and property owners by offering a centralized, transparent, and user-friendly system for listing, searching, and booking rental properties. Admins manage the approval process for owners, ensuring listings are authentic and verified.

SCENARIO

Scenario:

Abhi, a university student, is moving to a new city for his internship. He doesn't know anyone there and wants to find a verified rental property quickly without dealing with agents or fake listings.

Steps Abhi Follows on HouseHunt

1. Registration:

Abhi signs up as a renter by filling in his personal details.

2. Login:

He logs in using his email and password.

3. Explore Properties:

Abhi visits the “All Properties” section and uses filters like location and rent type.

4. View Property Details:

He clicks on a property card to view more images, details, and owner contact.

5. Book the Property:

Abhi fills in his name and phone number and submits a booking request.

6. View Booking History:

He checks the “Booking History” tab to confirm his booking was recorded.

7.Admin Management:

Meanwhile, the system administrator monitors all owners and approve them to post properties.The admin can also see all the properties and bookings.

Purpose

- Eliminate scams by ensuring owner verification before property listings go live.
- Allow renters to easily find and book properties online.
- Provide role-based access (Admin, Owner, Renter) with dedicated features.
- Improve the rental experience through a responsive, clean UI.
- Ensure platform transparency and reduce dependence on third parties.

Features

1. User Registration & Login

Secure role-based login for renters, owners, and admin

2. Admin Owner Approval

Admin manually approves owners before listings go live

3. Property Listing by Owners

Owners can upload property details with images

4. Property Search by Renters

Filterable property view for renters

5. Booking Request Submission

Renters can book properties directly with their details

6. Booking History Tracking

Renters can see all the bookings they've made

7. Role-Based Dashboard

Each user type sees a customized dashboard experience

3.ARCHITECTURE

Frontend Architecture (React.js)

The frontend of **ResolveNow** is developed using **React.js**, a powerful JavaScript library for building user interfaces. It follows a **component-based architecture**, allowing reusable and maintainable code. Major UI libraries like **Material UI** and **Bootstrap** are used to ensure responsiveness and modern design.

- **Axios** handles API communication between frontend and backend.
- Components are structured by role: UserDashboard, AdminPanel, OwnerDashboard, etc.

- Form validation, conditional rendering, and localStorage are used for session handling and role-specific access.

Backend Architecture (Node.js + Express.js)

The backend is built on **Node.js** with the **Express.js** framework, following a modular and RESTful API structure. It handles:

- **Routing** for user, admin, agent, and complaint-related endpoints
- **JWT-based Authentication** and middleware for access control
- **Multer** to handle file or image uploads
- Middleware like cors, body-parser, and custom error handlers for smooth request handling

Database Architecture (MongoDB + Mongoose)

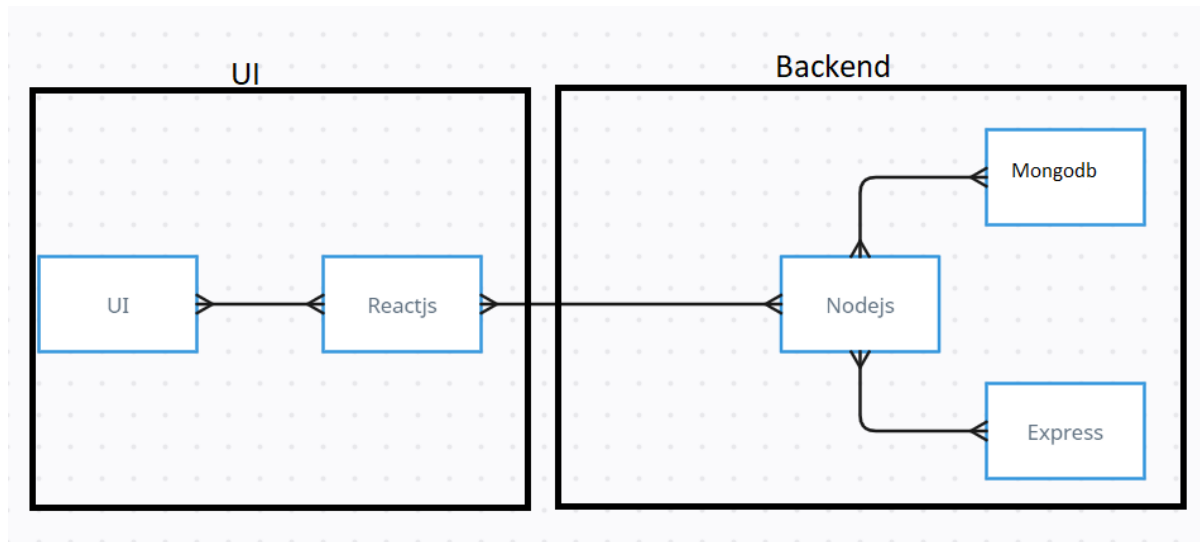
MongoDB is used as the NoSQL database to store application data. Mongoose ORM is used to define clear schemas and interact with the database efficiently.

Key Schemas:

- **User Schema:** Stores user details (name, email, password, role – user/admin/agent)
- **Property Schema:** Used by owners to post property listings
- **Booking Schema:** Stores all bookings made by users

Data is stored in collections and linked using ObjectIds, enabling relational behavior in a document-based model.

TECHNICAL ARCHITECTURE



The technical architecture of our househunt 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 agent, admin or 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, handling registration, login, property uploads, booking etc.

Together, the frontend and backend components, along with socket.io, Express.js, and MongoDB, form a comprehensive technical architecture for our HouseHunt online rental platform.

-

4. SETUP INSTRUCTIONS

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/>

✓ 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>

✓ **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 househunt rental app on your local machine. You can now proceed with further customization, development, and testing as needed.

5. FOLDER STRUCTURE

PROJECT STRUCTURE:

APPLICATION FLOW:

HouseHunt Rental Platform

1.Renter/User:

Features: Registration, Login, Search Properties, Book Property, View Booking History

- Can register and log in
- Can browse/search all properties
- Can book a property by submitting contact details
- Can view their booking history

2.Owner:

Features: Wait for Admin Approval, Post Properties, View Posted Properties

Registers as Owner (starts as "ungranted")

- Can log in after admin approval
- Once approved, can:
 - Upload properties with images
 - Set availability of property
 - Manage property listings

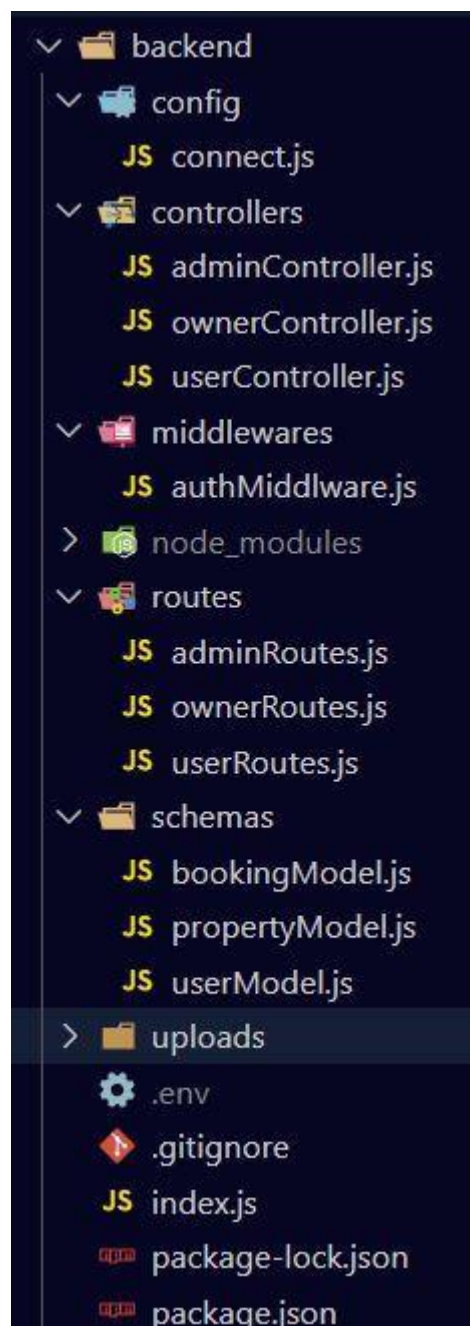
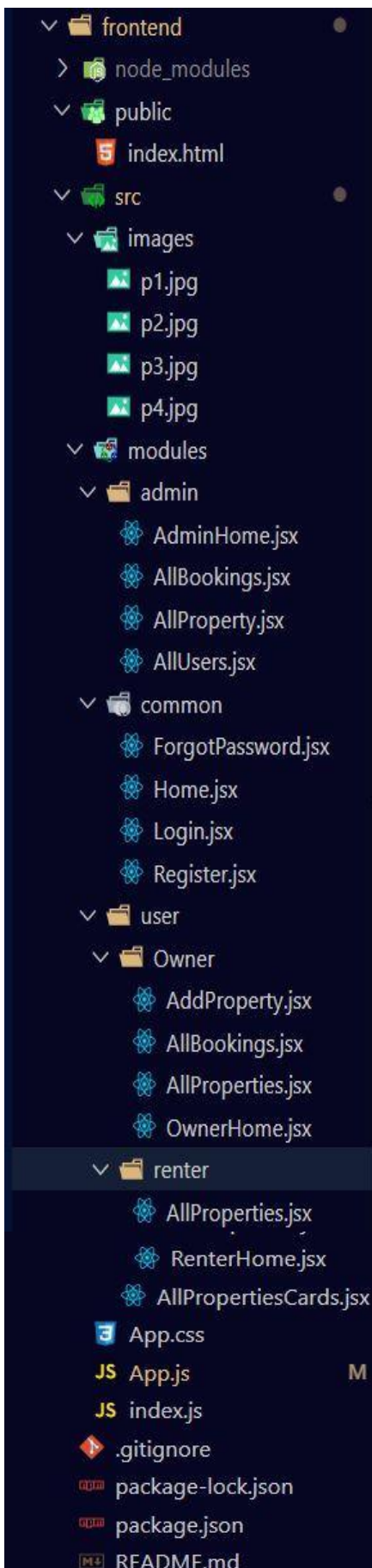
3.Admin:

Features: Admin Login, Approve Owners, Full Control Over Platform

Has special login credentials from .env

Can:

- Approve or reject owners
- View all registered users
- Manage the system without renting or uploading properties



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

To run the **HouseHunt** application locally, follow these steps:

Start the Frontend Server Copy code

```
cd frontend
```

```
npm install # Install all required frontend dependencies
```

```
npm start # Starts the React development server
```

The frontend will be available at:

<http://localhost:3000>

Start the Backend Server

Copy code

```
cd backend
```

```
npm install # Install all backend dependencies
```

```
node index.js # Starts the Node.js + Express server
```

- The backend will be running at: <http://localhost:5000>
- MongoDB will automatically connect using the configured URI (e.g., `mongodb://localhost:27017/complaintDB`)
- A success message like **"MongoDB connected"** or **"Database connected successfully"** will confirm a working connection in the terminal.

7. API DOCUMENTATION

Base URL

`http://localhost:5000/api`

Project Setup and Configuration

1. Create project folders and files:

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

- Client folders.
- Server folders

1. Install required tools and software:

For the backend to function well, we use the libraries mentioned in the prerequisites. Those libraries includes

- Node.js.
- MongoDB.

- Bcrypt
- Body-parser

Also, for the frontend we use the libraries such as

- React Js.
- Material UI
- Bootstrap
- Axios

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

```
frontend > package.json > ...
1  {
2    "name": "frontend",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "@emotion/react": "^11.11.1",
7      "@emotion/styled": "^11.11.0",
8      "@mui/icons-material": "^5.14.3",
9      "@mui/joy": "^5.0.0-beta.2",
10     "@mui/material": "^5.14.5",
11     "@testing-library/jest-dom": "^5.17.0",
12     "@testing-library/react": "^13.4.0",
13     "@testing-library/user-event": "^13.5.0",
14     "antd": "^5.8.3",
15     "axios": "^1.4.0",
16     "bootstrap": "^5.3.1",
17     "react": "^18.2.0",
18     "react-bootstrap": "^2.8.0",
19     "react-dom": "^18.2.0",
20     "react-router-dom": "^6.15.0",
21     "react-scripts": "5.0.1"
22   },
23   "scripts": {
24     "start": "react-scripts start",
25     "build": "react-scripts build",
26     "test": "react-scripts test",
27     "eject": "react-scripts eject"
28   },
29   "eslintConfig": {
30     "extends": [
31       "react-app",
32       "react-app/jest"
33     ]
34   },
35   "browserslist": {
36     "production": [
37       ">0.2%",
38       "not dead",
39       "not op_mini all"
40     ],
41     "development": [
```

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

```

backend > {} package.json > ...
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    >Debug
7    "scripts": {
8      "start": "nodemon index",
9      "test": "echo \"Error: no test specified\" && exit 1"
10   },
11   "keywords": [],
12   "author": "",
13   "license": "ISC",
14   "dependencies": {
15     "bcryptjs": "^2.4.3",
16     "cors": "^2.8.5",
17     "dotenv": "^16.3.1",
18     "express": "^4.18.2",
19     "jsonwebtoken": "^9.0.1",
20     "mongoose": "^7.4.3",
21     "multer": "^1.4.5-lts.1",
22     "nodemon": "^3.0.1"
23   }
24 }

```

Authentication Routes

User Registration (Signup)

Route: /api/user/register

Method: POST

Body (JSON):

```

{
  "name": "Tita",
  "email": "tita@gmail.com",
  "password": "tita123",
  "phone": "9876543210",
  "type": "Owner" // or "User"
}

```

Response (Success):

```

{
  "success": true,
  "message": "User Registered Successfully",
  "data": {
    "id": "64f2b....",
    "type": "Owner"
  }
}

```

2. User Login

Route: /api/user/login

Method: POST

Body (JSON):

```
{
  "email": "tita@gmail.com",
  "password": "tita123"
}
```

Response (Success):

```
{
  "success": true,
  "message": "Login Successful",
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "type": "Owner",
      "email": "tita@gmail.com"
    }
  }
}
```

Owner Routes

3. Add Property

Route: /api/owner/postproperty

Method: POST

Body: multipart/form-data with Authorization token

Fields:

propertyType
propertyAdType
propertyAddress
ownerContact
propertyAmt
additionalInfo
propertyImage (file)

Response:

```
{
  "success": true,
  "message": "New Property has been stored"
}
```

4. Update Property

Route: /api/owner/updateproperty/:id

Method: PUT

Body: multipart/form-data

Fields (any editable):

propertyType
propertyAdType
propertyAddress
ownerContact
propertyAmt
propertyImage (file)
additionalInfo

Response:

```
{  
  "success": true,  
  "message": "Property updated successfully"  
}
```

5. Delete Property

Route: /api/owner/deleteproperty/:id

Method: DELETE

Response:

```
{  
  "success": true,  
  "message": "The property is deleted"  
}
```

6. Get Owner's Properties

Route: /api/owner/getownerproperties

Method: GET

Headers: Authorization token

Response:

```
{  
  "success": true,  
  "data": [ ...properties... ]  
}
```

Booking Routes**7. Book a Property**

Route: /api/user/bookinghandle/:propertyId

Method: POST

Body (JSON):

```
{  
  "userDetails": {  
    "fullName": "Tita",  
    "phone": "9876543210"  
  },  
  "status": "pending",  
}
```

```
"ownerId": "65af..."
}
```

Response:

```
{
  "success": true,
  "message": "Booking request sent"
}
```

8. Owner - Get All Bookings

Route: /api/owner/getallbookings

Method: GET

Headers: Authorization token

Response:

```
{
  "success": true,
  "data": [ ...bookings... ]
}
```

9. Owner - Handle Booking Status

Route: /api/owner/handlebookingstatus

Method: POST

Body (JSON):

```
{
  "bookingId": "65bf...",
  "propertyId": "65cf...",
  "status": "booked" // or "rejected"
}
```

Response:

```
{
  "success": true,
  "message": "changed the status of property to booked"
}
```

8. AUTHENTICATION

The **HouseHunt** platform uses **JWT (JSON Web Token)** based authentication to securely verify and manage user sessions across different roles (renter, owner, admin).

How It Works:

1. User Login:

- On successful login, the server generates a **JWT token** containing the user's ID and role.
- The token is sent back to the client and stored in **localStorage**.

2. Token Format:

- Encodes user ID and role.
- Signed with a secret key defined in the .env file (JWT_SECRET).

3. Token Usage:

- The token is attached to all protected API requests in the Authorization header:

Authorization: Bearer <token>

4. Protected Routes:

- Backend uses middleware to verify the token before granting access to sensitive routes.
- Role-based access control is enforced
 - **Renter** can register book properties and view their own.
 - **Owners** can add ,update and delete properties.
 - **Admins** can manage users, manage bookings, and update statuses.

Backend Development

- **Set Up Project Structure:**
- Create a new directory for your project and set up a package.json file using npm init command.
- Install necessary dependencies such as Express.js, Mongoose, and other required packages.
- **Create Express.js Server:**
 - Set up an Express.js server to handle HTTP requests and serve API endpoints.
 - Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.
- **Define API Routes**
 - Created a routes/ folder.
 - Inside userRoutes.js, defined endpoints for registration, login, property management, bookings, etc.
 - Used express.Router() to organize routes clearly.
- **Create Controllers**
 - Created a controllers/ folder.
 - Each function (like registerController, loginController, bookingHandleController, etc.) handles the logic behind the route.
 - Separated logic for user, owner, and admin functionality.
- **Create Mongoose Models (Schemas)**
 - Added a schemas/ (or models/) folder.
 - Defined MongoDB schemas using Mongoose to represent:
 - Users (userModel.js)
 - Properties (propertyModel.js)
 - Bookings (bookingModel.js)
- **Setup Authentication Middleware**
 - Created authMiddleware.js to check JWT tokens.

- Verified users based on their role (Admin, Renter, Owner) before allowing route access.
- **Handle File Uploads (Property Images)**
 - Used multer to handle image uploads.
 - Created an uploads/ directory to store uploaded property images securely.
- **Configure Environment Variables**
 - Used .env to store sensitive information like:
 - MongoDB connection string
 - JWT secret key
 - Admin credentials
- **Connect to MongoDB**
 - Used Mongoose to connect backend to MongoDB via the server.js file.
 - Ensured all schemas were connected and working.
- **Test All APIs Using Postman**
- Registered users, logged in, uploaded properties, booked rentals, and verified role-based access via Postman.
- **Integrate with Frontend (React)**
 - Used axios on frontend to call backend APIs.
 - Sent JWT tokens in headers for protected routes like booking or property uploads.

Database Development Steps (MongoDB)

1. Design Schemas for User, Property, and Booking collections using Mongoose.
2. Define Relationships using ObjectId references to link users with properties and bookings.
3. Create Validation Rules (like required fields and default values).
4. Set Up Connection to MongoDB using Mongoose (mongoose.connect() with .env URI).
5. Test Data Insertion from backend (register user, add property, make booking).
6. Use MongoDB Compass to visually verify records and relationships.

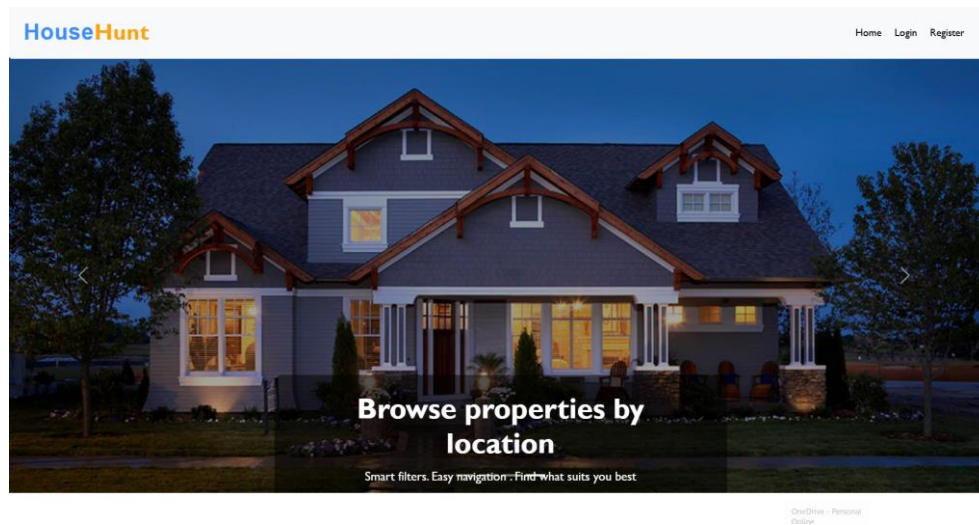
Frontend Development Steps (React.js + Bootstrap)

1. Create Page Structure using React components: Home, Register, Login, Dashboard, etc.
2. Build Forms for registration, login, property add/edit, and booking.
3. Connect to APIs using Axios with token-based headers for protected routes.
4. Design UI/UX using Bootstrap for layout, modals, tables, and cards.
5. Implement Conditional Routing for user/owner/admin dashboards using React Router.
6. Test Flows End-to-End, including property listing, filter, booking, and modal views.

9. USER INTERFACE

The **HouseHunt** user interface is built with **React.js**, styled using **Material UI** and **Bootstrap** to ensure a clean, modern, and responsive experience across devices.

HOME:



SIGNUP :

The screenshot displays the Sign Up page of the HouseHunt application. The layout is clean and centered. At the top, the 'HouseHunt' logo and navigation links ('Home', 'Login', 'Register') are visible. Below the header, there is a purple circular icon with a white lock symbol, followed by the text 'Sign up'. The form consists of four input fields: 'Renter Full Name/Owner Name' (containing 'Nikhil'), 'Email Address' (containing 'nikhilewar321@gmail.com'), 'Password' (masked with '*****'), and 'User Type' (a dropdown menu currently showing 'Owner'). A blue 'SIGN UP' button is positioned below the form fields. Below the button, there is a link that says 'Have an account? Sign In'. At the very bottom of the page, a footer bar contains the copyright notice '© 2025 Copyright: HouseHunt'.

LOGIN :

The screenshot shows the Sign In page of the HouseHunt application. The layout is consistent with the Sign Up page. At the top, the 'HouseHunt' logo and navigation links ('Home', 'Login', 'Register') are present. Below the header, there is a purple circular icon with a white lock symbol, followed by the text 'Sign In'. The form has two input fields: 'Email Address' (containing 'bhagi@gmail.com') and 'Password' (masked with '*****'). A blue 'SIGN IN' button is located below the form. Below the button, there are two links: 'forgot password? Click here' and 'Don't Have an account? Sign Up'. The footer bar at the bottom contains the copyright notice '© 2025 Copyright: HouseHunt'.

RENTER-DASHBOARD :

HouseHunt

Hi BhuvuLog Out

ALL PROPERTIES


BOOKING HISTORY

Filter By:

:Address

All Ad Types

All Types



Location:

Vijayawada

Property Type:

land/plot

Ad Type:

sale

Owner Contact:


6300742650

Availability:

Available

Property Amount:

Rs.900000



Location:

vijayawada

Property Type:

residential

Ad Type:

rent

Owner Contact:


9553198798

Availability:

Available

Property Amount:

Rs.6000



Location:

gudivada

Property Type:

land/plot

Ad Type:

sale

Owner Contact:


6304758617

Availability:

Available

Property Amount:

Rs.80000



Location:

hyderabad,madhapur

Property Type:

residential

Ad Type:

rent

Owner Contact:

6300742650

Availability:

Available

Property Amount:

Rs.8000

OWNER-STATUS :

HouseHunt

Hi AnilLog Out

ADD PROPERTY

ALL PROPERTIES

ALL BOOKINGS

Property ID	Property Type	Property Ad Type	Property Address	Owner Contact	Property Amt	Property Availability	Action
6860160ca059d25440a47346	land/plot	sale	Vijayawada	6300742650	900000	Available	<div>EditDelete</div>
68602a07a059d25440a473bf	residential	rent	vijayawada	9553198798	6000	Available	<div>EditDelete</div>
68602aa0a059d25440a473d2	land/plot	sale	gudivada	6304758617	80000	Available	<div>EditDelete</div>
68602b18a059d25440a473da	residential	rent	hyderabad,madhapur	6300742650	8000	Available	<div>EditDelete</div>

© 2025 Copyright: HouseHunt

ADMIN-DASHBAORD:

HouseHunt

Hi AdminLog Out

ALL USERS

ALL PROPERTIES

ALL BOOKINGS

User ID	Name	Email	Type	Granted (for Owners users only)	Actions
685ea70af32430b079a325b5	Bhuvana	22481A05KO@gmail.com	Owner	ungranted	<div>GRANTED</div>
685ea9b8f32430b079a325bb	Anil	anilju@uvarapu@gmail.com	Owner	granted	<div>UNGRANTED</div>
685eabadf32430b079a325c3	Nikhil	nikki@gmail.com	Renter		
685eabf4f32430b079a325cd	Maili	maili@gmail.com	Owner	granted	<div>UNGRANTED</div>
685ec4eb5c7e2ee79fb8284	Bhuvu	bhagi@gmail.com	Renter		
68602d19a059d25440a473ed	Nikhil	nikhileswar321@gmail.com	Owner	ungranted	<div>GRANTED</div>

© 2025 Copyright: HouseHunt

The UI of **HouseHunt** is built using **React.js**, styled with **Material UI** and **Bootstrap** for a responsive and clean layout.

Key Screens:

- **Login/Signup:** Simple, validated forms with role selection
- **Renter Dashboard:** can search properties, track booking status, book rental homes
- **Admin Panel:** approves owners, access bookings, manages entire platform
- **Owner Panel:** can add, update and delete properties, approves booking requests

10. TESTING

Testing Strategy

The testing approach for **HouseHunt** focused on verifying that all core functionalities worked as intended across different user roles (owner, renter, admin). The strategy included:

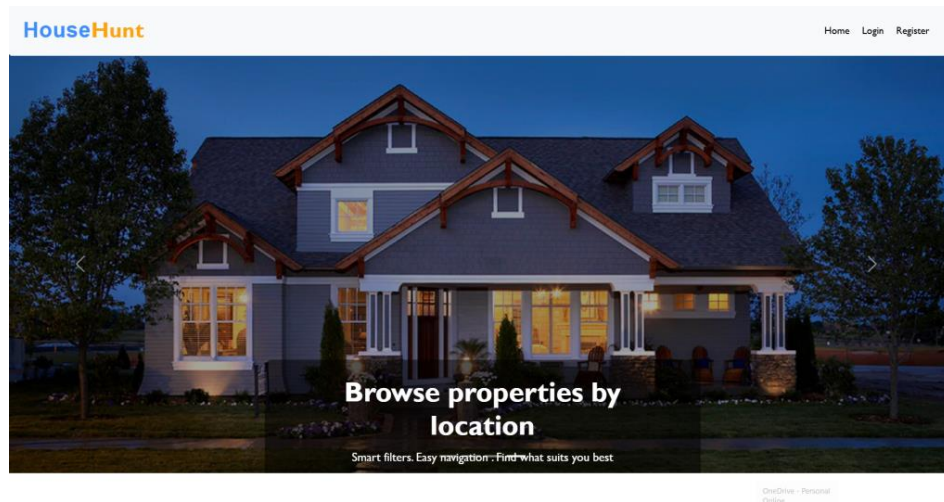
- **Manual Testing** of user flows such as:
 - Registration & Login
 - Add property as owner
 - Book property as a user
 - Check admin approval process
 - Test search, filter, and image display
- **Positive and Negative Scenarios** were covered to ensure:
 - Proper field validations
 - Unauthorized access is restricted
 - Functional correctness under real conditions

Tools Used

- **Postman** – For testing backend RESTful APIs (GET, POST, PATCH)
- **Browser DevTools** – For inspecting frontend behavior, component rendering, and network requests
- **React Developer Tools** – To test React components and props/state updates
- **VS Code Debugger** – To trace frontend function calls

11. SCREENSHOTS OR DEMO

HOME :



SIGNUP :

The screenshot displays the sign-up page of the HouseHunt website. The layout is clean and centered. At the top, the HouseHunt logo and navigation links (Home, Login, Register) are visible. Below the header, there is a purple circular icon with a white lock symbol, followed by the text "Sign up". The form consists of four input fields: "Renter Full Name/Owner Name" with the value "Nikhil", "Email Address" with the value "nikhileswar321@gmail.com", "Password" with masked characters "*****", and a "User Type" dropdown menu currently set to "Owner". A blue "SIGN UP" button is positioned below the form fields. Below the button, there is a link that says "Have an account? Sign In". At the very bottom of the page, a footer contains the copyright notice "© 2025 Copyright HouseHunt".

LOGIN :

RENTER-LOGIN

The screenshot shows the login page for renters on the HouseHunt website. The header with the HouseHunt logo and navigation links (Home, Login, Register) is at the top. The main content area features a purple circular icon with a white lock symbol and the text "Sign In". Below this, there are two input fields: "Email Address" with the value "bhag@gmail.com" and "Password" with masked characters "*****". A blue "SIGN IN" button is located below the password field. Underneath the button, there are two links: "forgot password? Click here" and "Don't Have an account? Sign Up". The footer at the bottom of the page displays the copyright notice "© 2025 Copyright HouseHunt".

RENTER -DASHBOARD :

HouseHunt

Hi Bhuva Log Out


ALL PROPERTIES

BOOKING HISTORY

Filter By: :Address

All Ad Types

All Types



Location:

Vijayawada

Property Type:

land/plot

Ad Type:

sale

Owner Contact:


6300742650

Availability:

Available

Property Amount:

Rs.900000



Location:

vijayawada

Property Type:

residential

Ad Type:

rent

Owner Contact:


9553198798

Availability:

Available

Property Amount:

Rs.6000



Location:

gudivada

Property Type:

land/plot

Ad Type:

sale

Owner Contact:


6304758617

Availability:

Available

Property Amount:

Rs.80000



Location:

hyderabad,madhapur

Property Type:

residential

Ad Type:

rent

Owner Contact:

6300742650

Availability:


Available

Property Amount:

Rs.8000

RENTER PROPERTY BOOKING:

Property Info



Owner Contact: 6300742650

Location: hyderabad,madhapur

Availability: Available

Property Type: residential

Property Amount: Rs.8000

Ad Type: rent

Additional Info:

Your Details to confirm booking


Full Name

Phone Number

Full Name

0

Book Property



Location:

Vijayawada

Property Type:

land/plot

Ad Type:

sale

Owner Contact:

6300742650

Availability:


Available

Property Amount:

Rs.900000

Get More Info of the Property

Get Info



Location:

vijayawada

Property Type:

residential

Ad Type:

rent

Owner Contact:

9553198798

Availability:


Available

Property Amount:

Rs.6000

Get More Info of the Property

Get Info



Location:

hyderabad,madhapur

Property Type:

residential

Ad Type:

rent

Owner Contact:

6300742650

Availability:

Available

Property Amount:

Rs.8000

Get More Info of the Property

Get Info

© 2025 Copyright: HouseHunt

ADMIN-LOGIN:

HouseHunt

HomeLoginRegister

Sign In

Email Address

admin@househunt.com

Password

SIGN IN

forgot password? Click here

Don't Have an account? Sign Up

© 2025 Copyright: HouseHunt

ADMIN-DASHBOARD:

HouseHunt

Hi AdminLog Out

ALL USERS

ALL PROPERTIES

ALL BOOKINGS

User ID	Name	Email	Type	Granted (for Owners users only)	Actions
685ea70af32430b079a325b5	Bhuvana	22481A05KO@gmail.com	Owner	ungranted	GRANTED
685ea9b8f32430b079a325bb	Anil	aniljujuvarapu@gmail.com	Owner	granted	UNGRANTED
685eabadf32430b079a325c3	Nikhil	nikki@gmail.com	Renter		
685eabf4f32430b079a325cd	Malli	malli@gmail.com	Owner	granted	UNGRANTED
685ec4eb5c7e26ee79fb9284	Bhuva	bhagi@gmail.com	Renter		
68602d19a059d25440a473ed	Nikhil	nikhleswar321@gmail.com	Owner	ungranted	GRANTED

© 2025 Copyright: HouseHunt

OWNER LOGIN:

HouseHunt

Home Login Register

Sign In

Email Address

aniljujuvarapu@gmail.com

Password

SIGN IN

forgot password? Click here

Don't Have an account? Sign Up

© 2025 Copyright HouseHunt

OWNER DASHBOARD:

HouseHunt

Hi Anil Log Out

ADD PROPERTY ALL PROPERTIES ALL BOOKINGS

Property type

Residential

Property Ad type

Rent

Property Full Address

Address

Property Images

Choose Files

No file chosen

Owner Contact No.

contact number

Property Amt.

0

Additional details for the Property

Submit form

© 2025 Copyright HouseHunt

OWNER EDITING PROPERTIES:

HouseHunt

Hi Anil

Log Out

ADD PROPERTY

ALL PROPERTIES

ALL BOOKINGS

Property ID	Property Type	Property Ad Type	Property Address	Owner Contact	Property Amt	Property Availability	Action
6860160ca059d25440a47346	land/plot	sale	Vijayawada	6300742650	900000	Available	<div>Edit</div> <div>Delete</div>
68602a07a059d25440a473bf	residential	rent	vijayawada	9553198798	6000	Available	<div>Edit</div> <div>Delete</div>
68602aa0a059d25440a473d2	land/plot	sale	gudivada	6304758617	80000	Available	<div>Edit</div> <div>Delete</div>
68602b18a059d25440a473da	residential	rent	hyderabad,madhapur	6300742650	8000	Available	<div>Edit</div> <div>Delete</div>

© 2025 Copyright: HouseHunt

Demo Link:

<https://drive.google.com/file/d/1uyKrGvL1whEhMzpobLie-ct5F8LD08wE/view?usp=drivesdk>

12. KNOWN ISSUES

- Add analytics dashboard for owners (views, bookings)
- Improve map integration to show nearby properties
- Add live chat between users and owners
- Develop mobile app version using React Native
- Generate downloadable receipts after booking

13. FUTURE ENHANCEMENTS

- Image preview not shown after choosing an image during edit
- Page does not auto-refresh after update/delete
- Network errors on slow connections
- No confirmation for successful booking form submission
- No logout button or session timeout handling