NAAN MUDHALVAN

House Rent App using MERN

Project Created by: K. Parvesh Kumar, S.Praveen, R.Sanjay, J.Arun, P.Saravanapandi

Project Created Date: 21/Nov/2024

College Code: 1127

College Name: ST. Peter's College of Engineering and Technology

Team Name:

ABSTRACT

House Rent App using MERN

The **House Rent App** is a modern web-based application designed to connect landlords and tenants on a unified digital platform. Built using the **MERN stack** (MongoDB, Express.js, React.js, and Node.js), the application provides scalable solutions for listing properties, facilitating communication, and managing secure rental payments.

With the rise in demand for digital solutions in the real estate sector, this app leverages cutting-edge technology to bridge the gap between landlords and tenants. Features include real-time property searches with filters, user dashboards for landlords and tenants, and integrated payment systems for secure transactions. This application not only ensures efficiency but also focuses on data privacy and user-friendly interfaces.

.

Project Description

Scenario 1: Renting an Apartment:

- **User Registration:** Alice, who is looking for a new apartment, downloads your house rent app and registers as a Renter. She provides her email and creates a password.
- **Browsing Properties:** Upon logging in, Alice is greeted with a dashboard showcasing available rental properties. She can see listings with detailed descriptions, photos, and rental information. She applies filters to narrow down her search, specifying her desired location, rent range, and the number of bedrooms.
- **Property Inquiry:** Alice finds an apartment she likes and clicks on it to get more information. She sees the property details and owner's contact information. Interested in renting, Alice fills out a small form with her details and sends it to the owner.

- **Booking Confirmation:** The owner receives Alice's inquiry and reviews her details. Satisfied, the owner approves Alice's booking request. Alice receives a notification that her booking is confirmed, and the status in her dashboard changes to "pending

owner confirmation."

- **Admin Approval (Background Process):** In the background, the admin reviews new owner registrations and approves legitimate users who want to add properties to the app.
- **Owner Management:** Bob, a property owner, signs up for an Owner account on the app and submits a request for approval. The admin verifies Bob's credentials and approves his Owner account.
- **Property Management:** With his Owner account approved, Bob can now add, edit, or delete properties in his account. He updates the status and availability of his properties based on their occupancy.
- **Platform Governance:** Meanwhile, the admin ensures that all users adhere to the platform's policies, terms of service, and privacy regulations. The admin monitors activities to maintain a safe and trustworthy environment for all users.
- **Transaction and Lease Agreement:** Once Alice's booking is confirmed, she and the owner negotiate the terms of the lease agreement through the app's messaging system. They finalize the rental contract and payment details within the app, ensuring transparency and security.
- **Move-in Process:** Alice successfully moves into her new apartment, marking the completion of the rental process facilitated by the house rent app.

This scenario highlights the main functionalities of your MERN-based house rent app, including user registration, property browsing, inquiry and booking process, admin approval, owner management, platform governance, and the overall rental transaction.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Background

The real estate rental market is often plagued with inefficiencies such as outdated listings, lack of secure communication channels, and cumbersome processes for rent payments. Traditional methods involve significant manual effort and are prone to errors, leading to a suboptimal experience for both landlords and tenants.

1.2 Objective

The primary objective of this project is to develop a feature-rich, scalable, and secure platform that simplifies the rental process. The platform aims to:

- Provide landlords with tools to manage properties efficiently.
- Offer tenants an easy-to-use interface for browsing properties and communicating with landlords.
- Ensure secure payment processing for rental transactions.

1.3 Scope

The House Rent App serves as a one-stop solution for landlords, tenants, and property managers. It supports:

- User registration and authentication.
- Property management for landlords.
- Advanced search capabilities for tenants.
- Secure communication and transactions.
- A scalable backend that can handle high user traffic.

CHAPTER 2

LITERATURE REVIEW

2.1 Existing Systems

Current rental platforms, such as Zillow and 99acres, often lack real-time updates, leading to outdated property information. Many platforms do not offer integrated payment systems or secure communication features, making them less efficient for end-users.

2.2 MERN Stack in Web Development

The MERN stack has gained popularity for its ability to build modern, dynamic web applications. React.js allows for a seamless user experience, while Node.js and Express.js handle robust backend services. MongoDB's NoSQL capabilities make it ideal for managing complex data such as property listings.

2.3Relevant Research

Studies on real estate technology reveal that digital platforms leveraging modern frameworks can significantly reduce inefficiencies in property management and tenant-landlord interactions. Research highlights the importance of secure payment systems and real-time communication features for enhancing user satisfaction.

CHAPTER 3

TECHNOLOGIES USED

3.1 Overview of System Architecture

The **House Rent App** system architecture is designed to provide a modular, scalable, and efficient solution for managing rental properties. It follows a **three-tier architecture** comprising the **Presentation Layer (Frontend)**, **Application Layer (Backend)**, and **Database Layer**, ensuring clear separation of concerns, scalability, and ease of maintenance.

3.2 Presentation Layer (Frontend)

The presentation layer is the interface that users interact with. Built with **React.js**, this layer ensures responsiveness, interactivity, and a seamless user experience.

- **Technologies Used**: React.js, Redux (for state management), CSS frameworks (e.g., Bootstrap or Material-UI).
- **Responsibilities**:
  - Rendering property listings and user dashboards.
  - Capturing user input (e.g., registration forms, property details).
  - Communicating with the backend via RESTful APIs.
- **Features**:
  - Search and filter functionality.
  - Interactive property cards with detailed descriptions and images.
  - Secure login and registration interfaces.

3.3 Application Layer (Backend)

The backend is responsible for processing user requests, managing business logic, and handling secure communication with the database. Developed using **Node.js** and **Express.js**, this layer ensures fast and efficient data processing.

- **Technologies Used**: Node.js, Express.js, JWT for authentication, and Bcrypt for password hashing.
- **Responsibilities**:
  - Processing API requests from the frontend.
  - Implementing CRUD operations for users, properties, and transactions.
  - Managing secure user sessions with JWT-based authentication.
  - Integrating third-party APIs (e.g., payment gateways like Stripe).
  - **Endpoints**:
  - /api/auth: User authentication (login, signup).
  - /api/properties: Manage property listings.
  - /api/users: Retrieve and update user profiles.
  - /api/payments: Handle rental payments.

3.4 Database Layer (MongoDB)

The database layer handles all data storage, retrieval, and management. Using **MongoDB**, a NoSQL database, allows flexibility in handling complex property and user data structures.

- **Technologies Used**: MongoDB (hosted on MongoDB Atlas).
- **Responsibilities**:
  - Storing property data, user profiles, and transaction records.
  - Providing efficient query mechanisms for real-time searches.
  - Ensuring data security and consistency with access control mechanisms.
- **Database Structure**:
  - **Users Collection**: Stores landlord and tenant data (e.g., name, email, hashed passwords).
  - **Properties Collection**: Stores property details (e.g., title, location, price, amenities).
  - **Transactions Collection**: Records payment details and statuses.

CHAPTER 4

Application Flow

4.1 Roles and Responsibilities:

The project has 2 type of user – Renter and Owner and other will be Admin which takes care to all the user. The roles and responsibilities of these two types of users can be inferred from the API endpoints defined in the code.

4.2 Here is a summary:

4.3 Renter/Tenent:

- Create an account and log in to the system using their email and password.
- They will be shown automatically all the properties in their dashboard.
- After clicking on the Get Info, all the information of the property and owner will come and small form will generate in which the renter needs to send his\her details.
- After that they can see their booking in booking section where the status of booking will be showing “pending”. It will be change by owner of the property.

4.4 Admin:

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

4.5 Owner:

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

CHAPTER 5

CODING

5.1 Front – End

1. **Index.html [**front-end/public/index.html**]**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

2. **AdminHome.jsx [**front-end/src/admin/AdminHome.jsx**]**

```
import React, { useState, useContext } from 'react'
import { Link } from 'react-router-dom';
import Navbar from 'react-bootstrap/Navbar';
import { Container, Nav } from 'react-bootstrap';
import { UserContext } from '../../App';
import PropTypes from 'prop-types';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';
import AllUsers from './AllUsers';
import AllProperty from './AllProperty';
import AllBookings from './AllBookings';
function CustomTabPanel(props) {
  const { children, value, index, ...other } = props;
  return (
    <div
      role="tabpanel"
```

```
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      {value === index && (
        <Box sx={{ p: 3 }}>
          <Typography>{children}</Typography>
        </Box>
      )}
    </div>
  );
}
CustomTabPanel.propTypes = {
  children: PropTypes.node,
  index: PropTypes.number.isRequired,
  value: PropTypes.number.isRequired,
};
function a11yProps(index) {
  return {
    id: `simple-tab-${index}`,
    'aria-controls': `simple-tabpanel-${index}`,
  };
}
const AdminHome = () => {
  const user = useContext(UserContext)
  const [value, setValue] = useState(0);

  const handleChange = (event, newValue) => {
    setValue(newValue);
  };
const handleLogOut = () => {
    localStorage.removeItem('token')
    localStorage.removeItem('user')
  }
  if (!user) {
    return null;;
  }
  return (
    <div>
      <Navbar expand="lg" className="bg-body-tertiary">
        <Container fluid>
          <Navbar.Brand><h2>RentEase</h2></Navbar.Brand>
          <Navbar.Toggle aria-controls="navbarScroll" />
          <Navbar.Collapse id="navbarScroll">
```

```
            <Nav
              className="me-auto my-2 my-lg-0"
              style={{ maxHeight: '100px' }}
              navbarScroll
            >
            </Nav>
            <Nav>
              <h5 className='mx-3'>Hi {user.userData.name}</h5>
              <Link onClick={handleLogOut} to={'/'}>Log Out</Link>
            </Nav>

          </Navbar.Collapse>
        </Container>
      </Navbar>

      <Box sx={{ width: '100%' }}>
        <Box sx={{ borderBottom: 1, borderColor: 'divider' }}>
          <Tabs value={value} onChange={handleChange} aria-label="basic tabs
example">
            <Tab label="All Users" {...a11yProps(0)} />
            <Tab label="All Properties" {...a11yProps(1)} />
            <Tab label="All Bookings" {...a11yProps(2)} />
          </Tabs>
        </Box>
        <CustomTabPanel value={value} index={0}>
          <AllUsers />
        </CustomTabPanel>
        <CustomTabPanel value={value} index={1}>
          <AllProperty />
        </CustomTabPanel>
        <CustomTabPanel value={value} index={2}>
          <AllBookings />
        </CustomTabPanel>
      </Box>
    </div>
  )
}

export default AdminHome
```

3. **AllBookings.jsx [**front-end/src/module/admin/AllBookings.jsx**]**

```
import { message } from 'antd';
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';

const AllBookings = () => {
  const [allBookings, setAllBookings] = useState([]);

  const getAllBooking = async () => {
    try {
      const response = await axios.get('http://localhost:8000/api/admin/getallbookings', {
        headers: { 'Authorization': `Bearer ${localStorage.getItem("token")}` }
      });

      if (response.data.success) {
        setAllBookings(response.data.data);
      } else {
        message.error(response.data.message);
      }
    } catch (error) {
      console.log(error);
    }
  };

  useEffect(() => {
    getAllBooking();
  }, []);

  return (
    <div>
      <TableContainer component={Paper}>
        <Table sx={{ minWidth: 650 }} aria-label="simple table">
          <TableHead>
            <TableRow>
              <TableCell>Booking ID</TableCell>
              <TableCell align="center">Owner ID</TableCell>
              <TableCell align="center">Property ID</TableCell>
              <TableCell align="center">Tenent ID</TableCell>
              <TableCell align="center">Tenent Name</TableCell>
              <TableCell align="center">Tenent Contact</TableCell>
              <TableCell align="center">Booking Status</TableCell>
            </TableRow>
          </TableHead>
          <TableBody>
```

```
            {allBookings.map((booking) => (
              <TableRow
                key={booking._id}
                sx={{ '&:last-child td, &:last-child th': { border: 0 } }}
              >
                <TableCell component="th" scope="row">
                  {booking._id}
                </TableCell>
                <TableCell align="center">{booking.ownerID}</TableCell>
                <TableCell align="center">{booking.propertyId}</TableCell>
                <TableCell align="center">{booking.userID}</TableCell>
                <TableCell align="center">{booking.userName}</TableCell>
                <TableCell align="center">{booking.phone}</TableCell>
                <TableCell align="center">{booking.bookingStatus}</TableCell>
              </TableRow>
            ))}
          </TableBody>
        </Table>
      </TableContainer>
    </div>
  );
};

export default AllBookings;
```

4. **AllUsers.jsx [**front-end/src/module/admin/ AllUsers.jsx**]**

```
import { message } from 'antd';
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import Button from '@mui/material/Button';
const AllUsers = () => {
   const [allUser, setAllUser] = useState([]);
   useEffect(() => {
      getAllUser();
   }, []);
   const getAllUser = async () => {
      try {
         const response = await
axios.get('http://localhost:8000/api/admin/getallusers', {
```

```
            headers: { 'Authorization': `Bearer
${localStorage.getItem("token")}` }
         });

         if (response.data.success) {
            setAllUser(response.data.data);
         } else {
            message.error(response.data.message);
         }
      } catch (error) {
         console.log(error);
      }
   };
   const handleStatus = async (userid, status) => {
      try {
         await axios.post('http://localhost:8000/api/admin/handlestatus', {
userid, status }, {
            headers: { 'Authorization': `Bearer
${localStorage.getItem("token")}` }
         }).then((res) => {
            if (res.data.success) {
               getAllUser();
            }
         });
      } catch (error) {
         console.log(error);
      }
   };
   return (
      <div>
         <TableContainer component={Paper}>
            <Table sx={{ minWidth: 650 }} aria-label="simple table">
               <TableHead>
                  <TableRow>
                     <TableCell>User ID</TableCell>
                     <TableCell align="center">Name</TableCell>
                     <TableCell align="center">Email</TableCell>
                     <TableCell align="center">Type</TableCell>
                     <TableCell align="center">Granted (for Owners users
only)</TableCell>
                     <TableCell align="center">Actions</TableCell>
                  </TableRow>
               </TableHead>
               <TableBody>
                  {allUser.map((user) => (
                     <TableRow
```

```
                        key={user._id}
                        sx={{ '&:last-child td, &:last-child th': { border: 0
} }}
                     >
                        <TableCell component="th" scope="row">
                           {user._id}
                        </TableCell>
                        <TableCell align="center">{user.name}</TableCell>
                        <TableCell align="center">{user.email}</TableCell>
                        <TableCell align="center">{user.type}</TableCell>
                        <TableCell align="center">{user.granted}</TableCell>
                        <TableCell align="center">
                           {user.type === 'Owner' && user.granted ===
'ungranted' ? (
                              <Button onClick={() => handleStatus(user._id,
'granted')} size='small' variant="contained" color="success">
                                 Granted
                              </Button>
                           ) : user.type === 'Owner' && user.granted ===
'granted' ? (
                              <Button onClick={() => handleStatus(user._id,
'ungranted')} size='small' variant="outlined" color="error">
                                 Ungranted
                              </Button>
                           ) : null}
                        </TableCell></TableRow>
                  ))}
               </TableBody>
            </Table>
         </TableContainer>
      </div>
   );
};

export default AllUsers;
```

5. **ForgotPassword.jsx [**front-end/src/module/common/ ForgotPassword.jsx**]**

```
import React, { useState } from 'react'
import { Container, Navbar, Nav } from 'react-bootstrap';
import { Link, useNavigate } from 'react-router-dom';
import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import TextField from '@mui/material/TextField';
import Grid from '@mui/material/Grid';
```

```
import Box from '@mui/material/Box';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import axios from 'axios';

const ForgotPassword = () => {
   const navigate = useNavigate()
   const [data, setData] = useState({
      email: '',
      password: '',
      confirmPassword: ''
   })

   const handleChange = (e) => {
      const { name, value } = e.target;
      setData({ ...data, [name]: value });
   };

   const handleSubmit = async (e) => {
      e.preventDefault();
      if (data.email === "" || data.password === "" || data.confirmPassword
=== "") {
         alert("Please fill all fields")
      } else {
         if (data.password === data.confirmPassword) {
            await axios.post("http://localhost:8000/api/user/forgotpassword",
data)
               .then((res) => {
                  if (res.data.success) {
                     alert('Your password has been changed!')
                     navigate('/login')
                  } else {
                     alert(res.data.message)
                  }
               })
               .catch((err) => {
                  if (err.response && err.response.status === 401) {
                     alert("User doesn't exist");
                  }
                  navigate("/register");
               });
         }

      }
```

```
   };
   return (
      <>
         <Navbar expand="lg" className="bg-body-tertiary">
            <Container fluid>
               <Navbar.Brand>
                  <Link to={'/'}>RentEase</Link>
               </Navbar.Brand>
               <Navbar.Toggle aria-controls="navbarScroll" />
               <Navbar.Collapse id="navbarScroll">
                  <Nav
                     className="me-auto my-2 my-lg-0"
                     style={{ maxHeight: '100px' }}
                     navbarScroll
                  >
                  </Nav>
                  <Nav>
                     <Link to={'/'}>Home</Link>
                     <Link to={'/login'}>Login</Link>
                     <Link to={'/register'}>Register</Link>
                  </Nav>

               </Navbar.Collapse>
            </Container>
         </Navbar>

         <Container component="main" maxWidth="xs">
            <Box
               sx={{
                  marginTop: 8,
                  display: 'flex',
                  flexDirection: 'column',
                  alignItems: 'center',
               }}
            >
               <Avatar sx={{ m: 1, bgcolor: 'secondary.main' }}>
                  <LockOutlinedIcon />
               </Avatar>
               <Typography component="h1" variant="h5">
                  Forgot Password?
               </Typography>
               <Box component="form" onSubmit={handleSubmit} noValidate sx={{
mt: 1 }}>
                  <TextField
                     margin="normal"
                     fullWidth
```

```
                    id="email"
                    label="Email Address"
                    name="email"
                    value={data.email}
                    onChange={handleChange}
                    autoComplete="email"
                    autoFocus
                 />
                 <TextField
                    margin="normal"
                    fullWidth
                    name="password"
                    value={data.password}
                    onChange={handleChange}
                    label="New Password"
                    type="password"
                    id="password"
                    autoComplete="current-password"
                 />
                 <TextField
                    margin="normal"
                    fullWidth
                    name="confirmPassword"
                    value={data.confirmPassword}
                    onChange={handleChange}
                    label="Confirm Password"
                    type="password"
                    id="confirmPassword"
                    autoComplete="current-password"
                 />
                 <Box mt={2}>
                    <Button
                       type="submit"
                       variant="contained"
                       sx={{ mt: 3, mb: 2 }}
                       style={{ width: '200px' }}
                    >
                       Change Password
                    </Button>
                 </Box>
                 <Grid container>
                    <Grid item>
                    </Grid>
                    <Grid item>Don't have an account?
                       <Link style={{ color: "red" }} to={'/register'}
variant="body2">
```

```
                        {" Sign Up"}
                     </Link>
                  </Grid>

               </Grid>
            </Box>
         </Box>
      </Container>
   </>
  )
}
export default ForgotPassword
```

6. **Home.jsx [**front-end/src/module/common/ Home.jsx**]**

```
import React, { useState } from 'react'
import { Link } from 'react-router-dom'
import Navbar from 'react-bootstrap/Navbar';
import { Container, Nav, Button } from 'react-bootstrap';
import Carousel from 'react-bootstrap/Carousel';
import p1 from '../../images/p1.jpg'
import p2 from '../../images/p2.jpg'
import p3 from '../../images/p3.jpg'
import p4 from '../../images/p4.jpg'
import AllPropertiesCards from '../user/AllPropertiesCards';
const Home = () => {
   const [index, setIndex] = useState(0);
   const handleSelect = (selectedIndex) => {
      setIndex(selectedIndex);
   };
   return (
      <>
         <Navbar expand="lg" className="bg-body-tertiary">
            <Container fluid>
               <Navbar.Brand><h2>RentEase</h2></Navbar.Brand>
               <Navbar.Toggle aria-controls="navbarScroll" />
               <Navbar.Collapse id="navbarScroll">
                  <Nav
                     className="me-auto my-2 my-lg-0"
                     style={{ maxHeight: '100px' }}
                     navbarScroll
                  >
                  </Nav>
                  <Nav>
                     <Link to={'/'}>Home</Link>
```

```
                  <Link to={'/login'}>Login</Link>
                  <Link to={'/register'}>Register</Link>
               </Nav>
            </Navbar.Collapse>
         </Container>
      </Navbar>
      <div className='home-body'>
         <Carousel activeIndex={index} onSelect={handleSelect}>
            <Carousel.Item>
               <img
                  src={p1}
                  alt="First slide"
               />
            </Carousel.Item>
            <Carousel.Item>
               <img
                  src={p2}
                  alt="Second slide"
               />
            </Carousel.Item>
            <Carousel.Item>
               <img
                  src={p3}
                  alt="Third slide"
               />
            </Carousel.Item>
            <Carousel.Item>
               <img
                  src={p4}
                  alt="Fourth slide"
               />
            </Carousel.Item>
         </Carousel>
      </div>
      <div className='property-content'>
         <div className='text-center'>
            <h1 className='m-1 p-5'>All Properties that may you look
for</h1>
            <p style={{fontSize: 15, fontWeight: 800}}>Want to post your
Property? <Link to={'/register'}><Button variant='outline-info'>Register as
Owner</Button></Link></p>
         </div>
         <Container>
            <AllPropertiesCards />
         </Container>
      </div>
```

```
      </>
   )
}

export default Home
```

7. **Login.jsx [**front-end/src/module/common/ Login.jsx**]**

```
import React, { useState } from 'react'
import { Link, useNavigate } from 'react-router-dom';
import Navbar from 'react-bootstrap/Navbar';
import { Container, Nav } from 'react-bootstrap';
import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import TextField from '@mui/material/TextField';
import Grid from '@mui/material/Grid';
import Box from '@mui/material/Box';
import LockOutlinedIcon from '@mui/icons-material/LockOutlined';
import Typography from '@mui/material/Typography';
import InputLabel from '@mui/material/InputLabel';
import MenuItem from '@mui/material/MenuItem';
import Select from '@mui/material/Select';
import axios from 'axios';
import { message } from 'antd';
const Register = () => {
  const navigate = useNavigate()
  const [data, setData] = useState({
    name: "",
    email: "",
    password: "",
    type: ""
  })

  const handleChange = (e) => {
    const { name, value } = e.target;
    setData({ ...data, [name]: value });
  };

  const handleSubmit = (e) => {
    e.preventDefault()
    if (!data?.name || !data?.email || !data?.password||! data?.type ) return
alert("Please fill all fields");
    else {
      axios.post('http://localhost:8000/api/user/register', data)
        .then((response) => {
```

```
        if (response.data.success) {
          message.success(response.data.message);
          navigate('/login')

        } else {
          message.error(response.data.message)
        }
      })
      .catch((error) => {
        console.log("Error", error);
      });
  }
};


return (
  <>
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container fluid>
        <Navbar.Brand><h2>RentEase</h2></Navbar.Brand>
        <Navbar.Toggle aria-controls="navbarScroll" />
        <Navbar.Collapse id="navbarScroll">
          <Nav
            className="me-auto my-2 my-lg-0"
            style={{ maxHeight: '100px' }}
            navbarScroll
          >
          </Nav>
          <Nav>
            <Link to={'/'}>Home</Link>
            <Link to={'/login'}>Login</Link>
            <Link to={'/register'}>Register</Link>
          </Nav>

        </Navbar.Collapse>
      </Container>
    </Navbar>

    <Container component="main" >
      <Box
        sx={{
          marginTop: 8,
          marginBottom: 4,
          display: 'flex',
          flexDirection: 'column',
          alignItems: 'center',
```

```
          }}
        >
          <Avatar sx={{ bgcolor: 'secondary.main' }}>
            <LockOutlinedIcon />
          </Avatar>
          <Typography component="h1" variant="h5">
            Sign up
          </Typography>
          <Box component="form" onSubmit={handleSubmit} noValidate>
            <TextField
              margin="normal"
              fullWidth
              id="name"
              label="Renter Full Name/Owner Name"
              name="name"
              value={data.name}
              onChange={handleChange}
              autoComplete="name"
              autoFocus
            />
            <TextField
              margin="normal"
              fullWidth
              id="email"
              label="Email Address"
              name="email"
              value={data.email}
              onChange={handleChange}
              autoComplete="email"
              autoFocus
            />
            <TextField
              margin="normal"
              fullWidth
              name="password"
              value={data.password}
              onChange={handleChange}
              label="Password"
              type="password"
              id="password"
              autoComplete="current-password"
            />
            <InputLabel id="demo-simple-select-label">User Type</InputLabel>
            <Select
              labelId="demo-simple-select-label"
              id="demo-simple-select"
```

```
              name='type'
              value={data.type}
              label="type"
              defaultValue="Select User"
              onChange={handleChange}
              style={{ width: '200px' }}
            >
              <MenuItem value={'Select User'} disabled>Select User</MenuItem>
              <MenuItem value={'Renter'}>Renter</MenuItem>
              <MenuItem value={"Owner"}>Owner</MenuItem>
            </Select>
            <Box mt={2}>
              <Button
                type="submit"
                variant="contained"
                style={{ width: '200px' }}
              >
                Sign Up
              </Button>
            </Box>
            <Grid container>
              <Grid item>Have an account?
                <Link style={{ color: "blue" }} to={'/login'} variant="body2">
                  {" Sign In"}
                </Link>
              </Grid>
            </Grid>
          </Box>
        </Box>
      </Container>
    </>
  )
}

export default Register
```

8. **AllPropertiesCards.jsx [**front-end/src /user/ AllPropertiesCards.jsx**]**

```
import axios from 'axios';
import React, { useState, useEffect } from 'react';
import { Button, Card, Modal, Carousel, Col, Form, InputGroup, Row } from
'react-bootstrap';
// import { Col, Form, Input, Row, message } from 'antd';
import { Link } from 'react-router-dom';
import { message } from 'antd';
const AllPropertiesCards = ({ loggedIn }) => {
   const [index, setIndex] = useState(0);
```

```
   const [show, setShow] = useState(false);
   const [allProperties, setAllProperties] = useState([]);
   const [filterPropertyType, setPropertyType] = useState('');
   const [filterPropertyAdType, setPropertyAdType] = useState('');
   const [filterPropertyAddress, setPropertyAddress] = useState('');
   const [propertyOpen, setPropertyOpen] = useState(null)
   const [userDetails, setUserDetails] = useState({
      fullName: '',
      phone: 0,
   })

   const handleChange = (e) => {
      const { name, value } = e.target;
      setUserDetails({ ...userDetails, [name]: value });
   };

   const handleClose = () => setShow(false);

   const handleShow = (propertyId) => {
      setPropertyOpen(propertyId)
      setShow(true)
   };

   const getAllProperties = async () => {
      try {
         const res = await
axios.get('http://localhost:8000/api/user/getAllProperties');
         setAllProperties(res.data.data);
      } catch (error) {
         console.log(error);
      }
   };

   const handleBooking = async (status, propertyId, ownerId) => {
      try {
         await
axios.post(`http://localhost:8000/api/user/bookinghandle/${propertyId}`, {
userDetails, status, ownerId }, {
            headers: {
               Authorization: `Bearer ${localStorage.getItem('token')}`
            }
         })
            .then((res) => {
               if (res.data.success) {
                  message.success(res.data.message)
                  handleClose()
```

```
            }
            else {
               message.error(res.data.message)
            }
         })
      } catch (error) {
         console.log(error);
      }
   }


   useEffect(() => {
      getAllProperties();
   }, []);



   const handleSelect = (selectedIndex) => {
      setIndex(selectedIndex);
   };

   const filteredProperties = allProperties
      .filter((property) => filterPropertyAddress === '' ||
property.propertyAddress.includes(filterPropertyAddress))
      .filter(
         (property) =>
            filterPropertyAdType === '' ||
            property.propertyAdType.toLowerCase().includes(filterPropertyAdTyp
e.toLowerCase())
      )
      .filter(
         (property) =>
            filterPropertyType === '' ||
            property.propertyType.toLowerCase().includes(filterPropertyType.to
LowerCase())
      );

   return (
      <>
         <div className=" mt-4 filter-container text-center">
            <p className="mt-3">Filter By: </p>
            <input
               type="text"
               placeholder=": Address"
               value={filterPropertyAddress}
               onChange={(e) => setPropertyAddress(e.target.value)}
```

```
            />
            <select value={filterPropertyAdType} onChange={(e) =>
setPropertyAdType(e.target.value)}>
               <option value="">All Ad Types</option>
               <option value="sale">Sale</option>
               <option value="rent">Rent</option>
            </select>
            <select value={filterPropertyType} onChange={(e) =>
setPropertyType(e.target.value)}>
               <option value="">All Types</option>
               <option value="commercial">Commercial</option>
               <option value="land/plot">land/Plot</option>
               <option value="residential">Residential</option>
            </select>
         </div>
         <div className="d-flex column mt-5">
            {filteredProperties && filteredProperties.length > 0 ? (
               filteredProperties.map((property) => (
                  <Card border="dark" key={property._id} style={{ width:
'18rem', marginLeft: 10 }}>
                     <Card.Body>
                        <Card.Title><img
src={`http://localhost:8000${property.propertyImage[0].path}`} alt='photos'
/></Card.Title>
                        <Card.Text>
                           <p style={{ fontWeight: 600 }} className='my-
1'>Location:</p> {property.propertyAddress} <br />
                           <p style={{ fontWeight: 600 }} className='my-
1'>Property Type:</p> {property.propertyType} <br />
                           <p style={{ fontWeight: 600 }} className='my-1'>Ad
Type:</p> {property.propertyAdType} <br />
                           {!loggedIn ? (
                              <>
                              </>
                           ) : (
                              <>
                                 <p style={{ fontWeight: 600 }} className='my-
1'>Owner Contact:</p> {property.ownerContact} <br />
                                 <p style={{ fontWeight: 600 }} className='my-
1'>Availabilty:</p> {property.isAvailable} <br />
                                 <p style={{ fontWeight: 600 }} className='my-
1'>Property Amount:</p> Rs.{property.propertyAmt}<br />
                              </>
                           )}
                        </Card.Text>
                        {
```

```
                          !loggedIn ? (<>
                             <p style={{ fontSize: 12, color: 'orange',
marginTop: 20 }}>For more details, click on get info</p>
                             <Link to={'/login'}>
                                <Button style={{ float: 'left' }}
variant="outline-dark">
                                   Get Info
                                </Button>
                             </Link></>
                          ) : (
                             <div>
                                {
    property.isAvailable === "Available" ? <><p style={{ float: 'left',
fontSize: 12, color: 'orange' }}>Get More Info of the Property</p>
  <Button onClick={() => handleShow(property._id)} style={{ float: 'right' }}
variant="outline-dark"> Get Info
                  </Button>
   <Modal show={show && propertyOpen === property._id} onHide={handleClose}>
    <Modal.Header closeButton>
    <Modal.Title>Property Info</Modal.Title>
                                          </Modal.Header>
                                          <Modal.Body>
      {property.propertyImage && property.propertyImage.length > 0 && (
     <Carousel activeIndex={index} onSelect={handleSelect}>
      {property.propertyImage.map((image, idx) => (
          <Carousel.Item key={idx}>
      <img src={`http://localhost:8001${image.path}`}
           alt=""    className="d-block w-100"
                                                      />
                                                   </Carousel.Item>
                                             ))}
                                          </Carousel>
                                       )}
                                       <div>
                          <div className="d-flex my-3">
                                     <div>

<p className='my-1'><b>Owner Contact:</b> {property.ownerContact} </p>
<p className='my-1'><b>Availabilty:</b> {property.isAvailable} </p>
<p className='my-1'><b>Property Amount: </b>Rs.{property.propertyAmt}</p>
                                  </div>
                        <div className="mx-4">
       <p className='my-1'><b>Location:</b> {property.propertyAddress} </p>
       <p className='my-1'><b>Property Type:</b> {property.propertyType} </p>
       <p className='my-1'><b>Ad Type: </b>{property.propertyAdType}</p>
                                                   </div>
```

```
                              </div>
  <p className='my-1'><b>Additional Info: </b>{property.additionalInfo}</p>
                                    </div>
                                          <hr />
                                          <div>
<span className='w-100'><h4><b>Your Details to confirm booking</b></h4></span>
                 <Form onSubmit={(e) => {
                                  e.preventDefault();
                     handleBooking('pending', property._id, property.ownerId)
                                           }}>
                              <Row className="mb-3">
                              <Form.Group as={Col} md="6">
                                   <Form.Label>Full Name</Form.Label>
                                     <InputGroup hasValidation>
<Form.Control type="text" placeholder="Full Name" aria-
describedby="inputGroupPrepend" required  name='fullName'
value={userDetails.fullName}   onChange={handleChange}/>
                                                              </InputGroup>
                                                          </Form.Group>
                                       <Form.Group as={Col} md="6">
                            <Form.Label>Phone Number</Form.Label>
                           <InputGroup hasValidation>
      <Form.Control type="number" placeholder="Phone Number"
   aria-describedby="inputGroupPrepend" required
name='phone' value={userDetails.phone} onChange={handleChange} />
                                                              </InputGroup>
                                                          </Form.Group>
                                                      </Row>
 <Button type='submit' variant="secondary">Book Property</Button>
                  </Form>
                            </div>
              </Modal.Body>
        </Modal></> : <p>Not Available</p>    </div>
                         )
                       }
                    </Card.Body>
                  </Card>
              ))
           ) : (
              <p>No Properties available at the moment.</p>
           )}
        </div>
      </>
   );};
export default AllPropertiesCards;
```

9. **App.css [**front-end/src / App.css**]**

```
@import "../node_modules/bootstrap/dist/css/bootstrap.min.css";

* {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
   font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-
serif;

}

html,
body {
   width: 100%;
   height: 100%;
}

a {
   color: black;
   text-decoration: none;
   margin-right: 20px;
}

.App {
   display: flex;
   flex-direction: column;
   min-height: 100vh;
}

.content {
   flex: 1;
}

.home-body {
   height: 81.5vh;
   overflow: hidden;
}

.home-body .carousel {
   max-height: 100%;
}

.home-body .carousel .carousel-item {
   height: 100%;
```

```
}

.home-body .carousel .carousel-item img {
   width: 100%;
   height: 100%;
   object-fit: contain;
   background-size: cover;
}

.property-content {
   display: flex;
   flex-wrap: wrap;
   justify-content: center;
}

.content-home {
   position: absolute;
   top: 30%;
   left: 5%;
}

.content-home p {
   font-size: 42px;
   font-weight: 800;
   color: rgb(102, 62, 9);
}

.filter-container {
   width: 100%;
   height: 10vh;
   display: flex;
   align-items: center;
   justify-content: center;
   margin-bottom: 10px;
}

.filter-container input,
select {
   border: 1px solid black;
   padding: 10px;
   margin: 10px;
   border-radius: 5px;
}

.column {
   width: 100%;
```

```
    display: flex;
    flex-wrap: wrap;
}

.card {
    margin-right: 20px;
    margin-bottom: 15px;
}

.card .card-body .card-title img {
    width: 100%;
    height: 150px;
}

.chat-container {
    display: flex;
    flex-direction: column;
    height: 100%;
}

.chat-container h1 {
    font-size: 22px;
    text-align: center;
    margin-bottom: 20px;
}


.message-window {
    /* flex: 1; */
    overflow-x: auto;
    padding: 10px;
    width: 100%;
    max-height: 200px;
    height: 100px;
    border: 1px solid lightgray;
    border-radius: 5px;
}

.message {
    background-color: #f5f5f5;
    padding: 5px;
    margin-bottom: 10px;
}

.input-container {
    display: flex;
```

```
   align-items: center;
   padding: 12px;
}

.input-container textarea {
   width: 25px;
   flex: 1;
   padding: 5px;
   margin-right: 5px;
   border: none;
   border-radius: 5px;
   background-color: rgb(250, 247, 247);
}

.input-container button {
   padding: 5px 10px;
   color: white;
   font-size: 12px;
   border: none;
   width: 40px;
   height: 40px;
   border-radius: 50%;
}

@media screen and (max-width: 480px) {
   .chat-container {
      height: auto;
   }
}
```

1. **App.js [**front-end/src/ App.js**]**

```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import "./App.css";
import Home from "./modules/common/Home";
import Login from "./modules/common/Login";
import Register from "./modules/common/Register";
import ForgotPassword from "./modules/common/ForgotPassword";
import { createContext, useEffect, useState } from "react";
import AdminHome from "./modules/admin/AdminHome";
import OwnerHome from "./modules/user/Owner/OwnerHome";
import RenterHome from "./modules/user/renter/RenterHome";
export const UserContext = createContext();

function App() {
```

```
  const date = new Date().getFullYear();
  const [userData, setUserData] = useState();
const [userLoggedIn, setUserLoggedIn] = useState(false)
  const getData = async () => {
    try {
      const user = await JSON.parse(localStorage.getItem("user"));
      if (user && user !== undefined) {
        setUserData(user);
        setUserLoggedIn(true)
      }
    } catch (error) {
      console.log(error);
    }
  };

  useEffect(() => {
    getData();
  }, []);

  // const userLoggedIn = !!localStorage.getItem("user");
  return (
    <UserContext.Provider value={{userData, userLoggedIn}}>
      <div className="App">
        <Router>
          <div className="content">
            <Routes>
              <Route path="/" element={<Home />} />
              <Route path="/login" element={<Login />} />
              <Route path="/register" element={<Register />} />
              <Route path="/forgotpassword" element={<ForgotPassword />} />
              {userLoggedIn ? (
                <>
                  <Route path="/adminhome" element={<AdminHome />} />
                  <Route path="/ownerhome" element={<OwnerHome />} />
                  <Route path="/renterhome" element={<RenterHome />} />
                </>
              ) : (
                <Route path="/login" element={<Login />} />
              )}
            </Routes>
          </div>
          <footer className="bg-light text-center text-lg-start">
            <div className="text-center p-3">
              © {date} Copyright: RentEase
            </div>
          </footer>
```

```
        </Router>
      </div>
    </UserContext.Provider>
  );
}
export default App;
```

2. **Index.js [**front-end/src/ Index.js**]**

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

5.2 Black – End

1. **connect.js [**back-end/config/ connect.js**]**

```
const mongoose = require('mongoose');
const connectionOfDb = () => {
  const dbURI = 'mongodb://127.0.0.1:27017/test';

  mongoose
    .connect(dbURI, { useNewUrlParser: true, useUnifiedTopology: true })
    .then(() => {
      console.log('Connected to MongoDB');
    })
    .catch((err) => {
      console.error(`Could not connect to MongoDB: ${err}`);
      process.exit(1);  // Ensure the app exits on failure to connect to
MongoDB
    });
};

module.exports = connectionOfDb;
```

2. **ownerController.js [**bank-end/ controller/ ownerController.js**]**

```
const bookingSchema = require("../schemas/bookingModel");
const propertySchema = require("../schemas/propertyModel");
const userSchema = require("../schemas/userModel");

//////////adding property by owner////////
const addPropertyController = async (req, res) => {
  try {
    let images = [];
    if (req.files) {
      images = req.files.map((file) => ({
        filename: file.filename,
        path: `/uploads/${file.filename}`,
      }));
    }

    const user = await userSchema.findById({ _id: req.body.userId });

    const newPropertyData = new propertySchema({
      ...req.body,
      propertyImage: images,
      ownerId: user._id,
      ownerName: user.name,
      isAvailable: "Available",
    });

    await newPropertyData.save();

    return res.status(200).send({
      success: true,
      message: "New Property has been stored",
    });
  } catch (error) {
    console.log("Error in get All Users Controller ", error);
  }
};

//////////all properties of owner////////
const getAllOwnerPropertiesController = async (req, res) => {
  const { userId } = req.body;
  try {
    const getAllProperties = await propertySchema.find();
    const updatedProperties = getAllProperties.filter(
      (property) => property.ownerId.toString() === userId
    );
    return res.status(200).send({
      success: true,
```

```
      data: updatedProperties,
    });
  } catch (error) {
    console.error(error);
    return res
      .status(500)
      .send({ message: "Internal server error", success: false });
  }
};

//////delete the property by owner/////
const deletePropertyController = async (req, res) => {
  const propertyId = req.params.propertyid;
  try {
    await propertySchema.findByIdAndDelete({
      _id: propertyId,
    });

    return res.status(200).send({
      success: true,
      message: "The property is deleted",
    });
  } catch (error) {
    console.error(error);
    return res
      .status(500)
      .send({ message: "Internal server error", success: false });
  }
};

//////updating the property//////////////
const updatePropertyController = async (req, res) => {
  const { propertyid } = req.params;
  console.log(req.body);
  try {
    const property = await propertySchema.findByIdAndUpdate(
      { _id: propertyid },
      {
        ...req.body,
        ownerId: req.body.userId,
      },
      { new: true }
    );

    return res.status(200).send({
      success: true,
```

```
      message: "Property updated successfully.",
    });
  } catch (error) {
    console.error("Error updating property:", error);
    return res.status(500).json({
      success: false,
      message: "Failed to update property.",
    });
  }
};

const getAllBookingsController = async (req, res) => {
  const { userId } = req.body;
  try {
    const getAllBookings = await bookingSchema.find();
    const updatedBookings = getAllBookings.filter(
      (booking) => booking.ownerID.toString() === userId
    );
    return res.status(200).send({
      success: true,
      data: updatedBookings,
    });
  } catch (error) {
    console.error(error);
    return res
      .status(500)
      .send({ message: "Internal server error", success: false });
  }
};

//////////handle bookings status//////////////
const handleAllBookingstatusController = async (req, res) => {
  const { bookingId, propertyId, status } = req.body;
  try {
    const booking = await bookingSchema.findByIdAndUpdate(
      { _id: bookingId },
      {
        bookingStatus: status,
      },
      {
        new: true,
      }
    );

    const property = await propertySchema.findByIdAndUpdate(
      { _id: propertyId },
```

```
      {
        isAvailable: status === 'booked' ? 'Unavailable' : 'Available',
      },
      { new: true }
    );

    return res.status(200).send({
      success: true,
      message: `changed the status of property to ${status}`,
    });
  } catch (error) {
    console.error(error);
    return res
      .status(500)
      .send({ message: "Internal server error", success: false });
  }
};
module.exports = {
  addPropertyController,
  getAllOwnerPropertiesController,
  deletePropertyController,
  updatePropertyController,
  getAllBookingsController,
  handleAllBookingstatusController,};
```

3. **userController.js [**bank-end/ controller/ userController.js**]**

```
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const userSchema = require("../schemas/userModel");
const propertySchema = require("../schemas/propertyModel");
const bookingSchema = require("../schemas/bookingModel");

// Ensure JWT_KEY is loaded
if (!process.env.JWT_KEY) {
  throw new Error("JWT_KEY is missing in environment variables. Please set it
in your .env file.");
}

////////// Register Controller //////////
const registerController = async (req, res) => {
  try {
    let granted = "";
    const existsUser = await userSchema.findOne({ email: req.body.email });
    if (existsUser) {
      return res
```

```
        .status(200)
        .send({ message: "User already exists", success: false });
    }
    const password = req.body.password;
    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(password, salt);
    req.body.password = hashedPassword;

    if (req.body.type === "Owner") {
      granted = "ungranted";
      const newUser = new userSchema({ ...req.body, granted });
      await newUser.save();
    } else {
      const newUser = new userSchema(req.body);
      await newUser.save();
    }

    return res.status(201).send({ message: "Register Success", success: true
});
  } catch (error) {
    console.log(error);
    return res
      .status(500)
      .send({ success: false, message: `${error.message}` });
  }
};

////////// Login Controller //////////
const loginController = async (req, res) => {
  try {
    const user = await userSchema.findOne({ email: req.body.email });
    if (!user) {
      return res
        .status(200)
        .send({ message: "User not found", success: false });
    }
    const isMatch = await bcrypt.compare(req.body.password, user.password);
    if (!isMatch) {
      return res
        .status(200)
        .send({ message: "Invalid email or password", success: false });
    }
    const token = jwt.sign({ id: user._id }, process.env.JWT_KEY, {
      expiresIn: "1d",
    });
```

```
    user.password = undefined;
    return res.status(200).send({
      message: "Login success successfully",
      success: true,
      token,
      user: user,
    });
  } catch (error) {
    console.log(error);
    return res
      .status(500)
      .send({ success: false, message: `${error.message}` });
  }
};

////////// Forgot Password Controller //////////
const forgotPasswordController = async (req, res) => {
  try {
    const { email, password } = req.body;

    // Hash the new password
    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(password, salt);

    const updatedUser = await userSchema.findOneAndUpdate(
      { email },
      { password: hashedPassword },
      { new: true }
    );

    if (!updatedUser) {
      return res
        .status(200)
        .send({ message: "User not found", success: false });
    }

    await updatedUser.save();
    return res.status(200).send({
      message: "Password changed successfully",
      success: true,
    });
  } catch (error) {
    console.log(error);
    return res
      .status(500)
      .send({ success: false, message: `${error.message}` });
```

```
  }
};

////////// Auth Controller //////////
const authController = async (req, res) => {
  console.log(req.body);
  try {
    const user = await userSchema.findOne({ _id: req.body.userId });
    console.log(user);
    if (!user) {
      return res
        .status(200)
        .send({ message: "User not found", success: false });
    } else {
      return res.status(200).send({
        success: true,
        data: user,
      });
    }
  } catch (error) {
    console.log(error);
    return res
      .status(500)
      .send({ message: "Auth error", success: false, error });
  }
};

////////// Get All Properties Controller //////////
const getAllPropertiesController = async (req, res) => {
  try {
    const allProperties = await propertySchema.find({});
    if (!allProperties) {
      throw new Error("No properties available");
    } else {
      res.status(200).send({ success: true, data: allProperties });
    }
  } catch (error) {
    console.log(error);
    return res
      .status(500)
      .send({ message: "Error fetching properties", success: false, error });
  }
};

////////// Booking Handle Controller //////////
const bookingHandleController = async (req, res) => {
```

```
  const { propertyid } = req.params;
  const { userDetails, status, userId, ownerId } = req.body;

  try {
    const booking = new bookingSchema({
      propertyId: propertyid,
      userID: userId,
      ownerID: ownerId,
      userName: userDetails.fullName,
      phone: userDetails.phone,
      bookingStatus: status,
    });

    await booking.save();

    return res
      .status(200)
      .send({ success: true, message: "Booking status updated" });
  } catch (error) {
    console.error("Error handling booking:", error);
    return res
      .status(500)
      .send({ success: false, message: "Error handling booking" });
  }
};

////////// Get All Bookings Controller //////////
const getAllBookingsController = async (req, res) => {
  const { userId } = req.body;
  try {
    const getAllBookings = await bookingSchema.find();
    const updatedBookings = getAllBookings.filter(
      (booking) => booking.userID.toString() === userId
    );
    return res.status(200).send({
      success: true,
      data: updatedBookings,
    });
  } catch (error) {
    console.error(error);
    return res
      .status(500)
      .send({ message: "Internal server error", success: false });
  }
};
module.exports = {
```

```
  registerController,
  loginController,
  forgotPasswordController,
  authController,
  getAllPropertiesController,
  bookingHandleController,
  getAllBookingsController,};
```

4. **authMiddlware.js [**bank-end/ middlewares/ authMiddlware.js**]**

```
const jwt = require("jsonwebtoken");

module.exports = async (req, res, next) => {
  try {
    const authorizationHeader = req.headers["authorization"];
    if (!authorizationHeader) {
      return res
        .status(401)
        .send({ message: "Authorization header missing", success: false });
    }

    const token = req.headers["authorization"].split(" ")[1];
    jwt.verify(token, process.env.JWT_KEY, (err, decode) => {
      if (err) {
        return res
          .status(200)
          .send({ message: "Token is not valid", success: false });
      } else {
        req.body.userId = decode.id;
        next();
      }
    });
  } catch (error) {
    console.error(error); // Handle or log the error appropriately
    res.status(500).send({ message: "Internal server error", success: false
});
  }
};
```

5. **adminRoutes.js [**bank-end/ routes/ adminRoutes.js**]**

```
const express = require("express");
const authMiddlware = require("../middlewares/authMiddlware");
```

```
const { getAllUsersController, handleStatusController,
getAllPropertiesController, getAllBookingsController } =
require("../controllers/adminController");
const router = express.Router()
router.get('/getallusers', authMiddlware, getAllUsersController)
router.post('/handlestatus', authMiddlware, handleStatusController)
router.get('/getallproperties', authMiddlware, getAllPropertiesController)
router.get('/getallbookings', authMiddlware, getAllBookingsController)
module.exports = router
```

6. **userRoutes.js [**bank-end/ routes / userRoutes.js**]**

```
const express = require("express");
const authMiddleware = require("../middlewares/authMiddlware");
const {
  registerController,
  loginController,
  forgotPasswordController,
  authController,
  getAllPropertiesController,
  bookingHandleController,
  getAllBookingsController,
} = require("../controllers/userController");
const router = express.Router();
router.post("/register", registerController);
router.post("/login", loginController);
router.post("/forgotpassword", forgotPasswordController);
router.get('/getAllProperties', getAllPropertiesController)
router.post("/getuserdata", authMiddleware, authController);
router.post("/bookinghandle/:propertyid", authMiddleware,
bookingHandleController);
router.get('/getallbookings', authMiddleware, getAllBookingsController)
module.exports = router;
```

7. **bookingModel.js [**bank-end/ schemas/ bookingModel.js**]**

```
const mongoose = require("mongoose");
const bookingModel = mongoose.Schema(
  {
    propertId: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "propertyschema",
    },
    ownerID: {
      type: mongoose.Schema.Types.ObjectId,
```

```
      ref: "user",
    },
    userID: {
      type: mongoose.Schema.Types.ObjectId,
      ref: "user",
    },
    userName: {
      type: String,
      required: [true, "Please provide a User Name"],
    },
    phone: {
      type: Number,
      required: [true, "Please provide a Phone Number"],
    },
    bookingStatus: {
      type: String,
      required: [true, "Please provide a booking Type"],
    },
  },
  {
    strict: false,
  }
);
const bookingSchema = mongoose.model("bookingschema", bookingModel);
module.exports = bookingSchema;
```

8. **propertyModel.js [**bank-end/ schemas/ propertyModel.js**]**

```
const mongoose = require('mongoose')
const propertyModel = mongoose.Schema({
   ownerId:{
      type: mongoose.Schema.Types.ObjectId,
      ref: 'user'
   },
   propertyType:{
      type:String,
      required:[true,'Please provide a Property Type']
   },
   propertyAdType:{
      type: String,
      required:[true,'Please provide a Property Ad Type']
   },
   propertyAddress:{
      type: String,
      required:[true,"Please Provide an Address"]
   },
```

```
   ownerContact:{
      type: Number,
      required: [true, 'Please provide owner contact']
   },
   propertyAmt:{
      type :Number ,
      default: 0,
   },
   propertyImage: {
      type: Object
   },
   additionalInfo:{
      type: String,
   },
   ownerName: {
      type: String,
   }
},{
   strict: false,
})
const propertySchema = mongoose.model('propertyschema', propertyModel)
module.exports = propertySchema
```

9. **userModel.js [**bank-end/ schemas/ userModel.js**]**

```
const mongoose = require("mongoose");
const userModel = mongoose.Schema({
  name: {
    type: String,
    required: [true, "Name is required"],
    set: function (value) {
      return value.charAt(0).toUpperCase() + value.slice(1);
    },
  },
  email: {
    type: String,
    required: [true, "email is required"],
  },
  password: {
    type: String,
    required: [true, "password is required"],
  },
  type: {
    type: String,
    required: [true, "type is required"],
  },
```

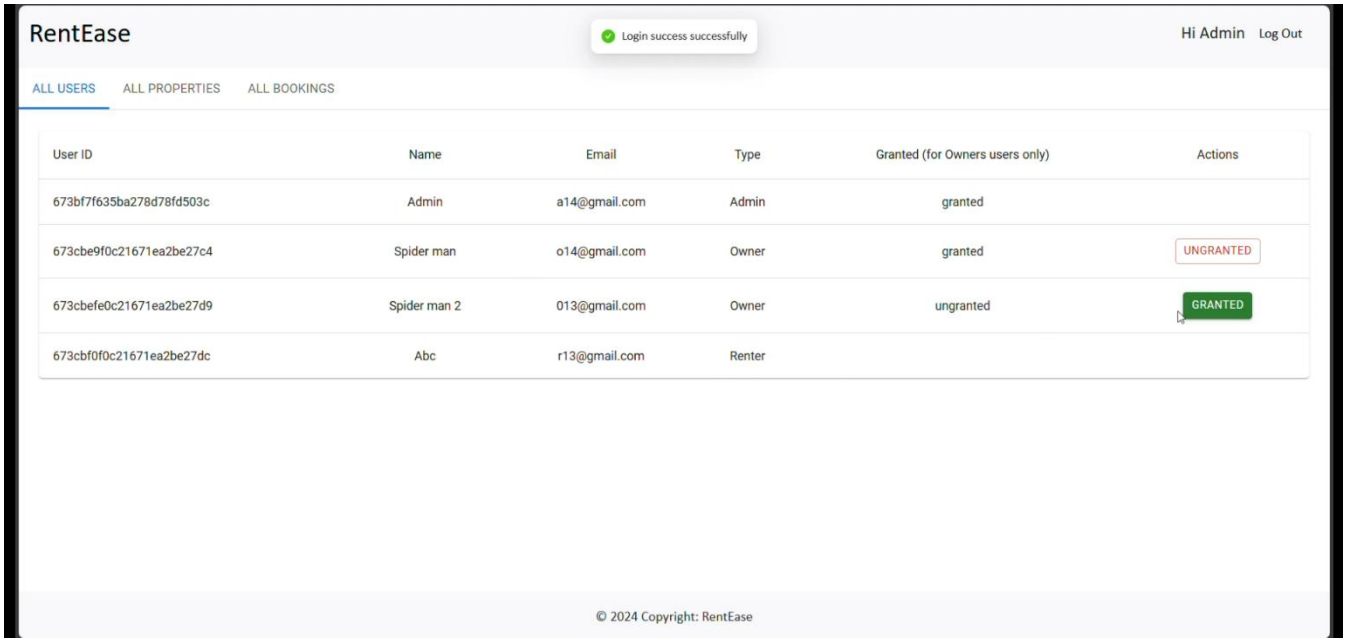
```
},{
   strict: false,
});
const userSchema = mongoose.model("user", userModel);
module.exports = userSchema;
```

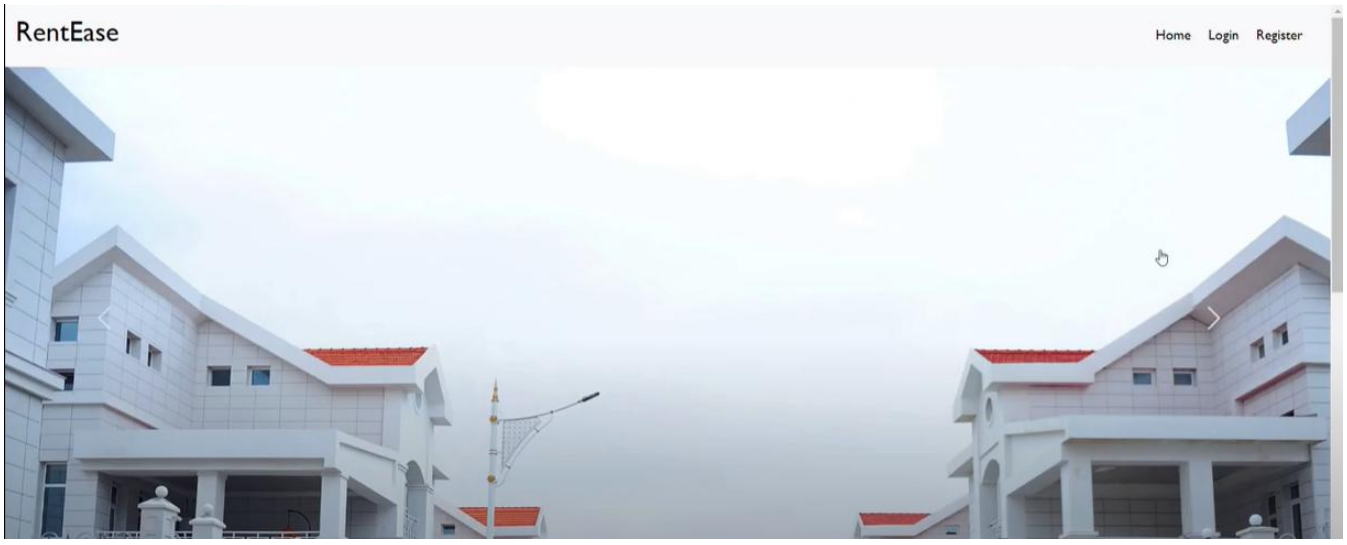
10. index.js [bank-end / index.js]

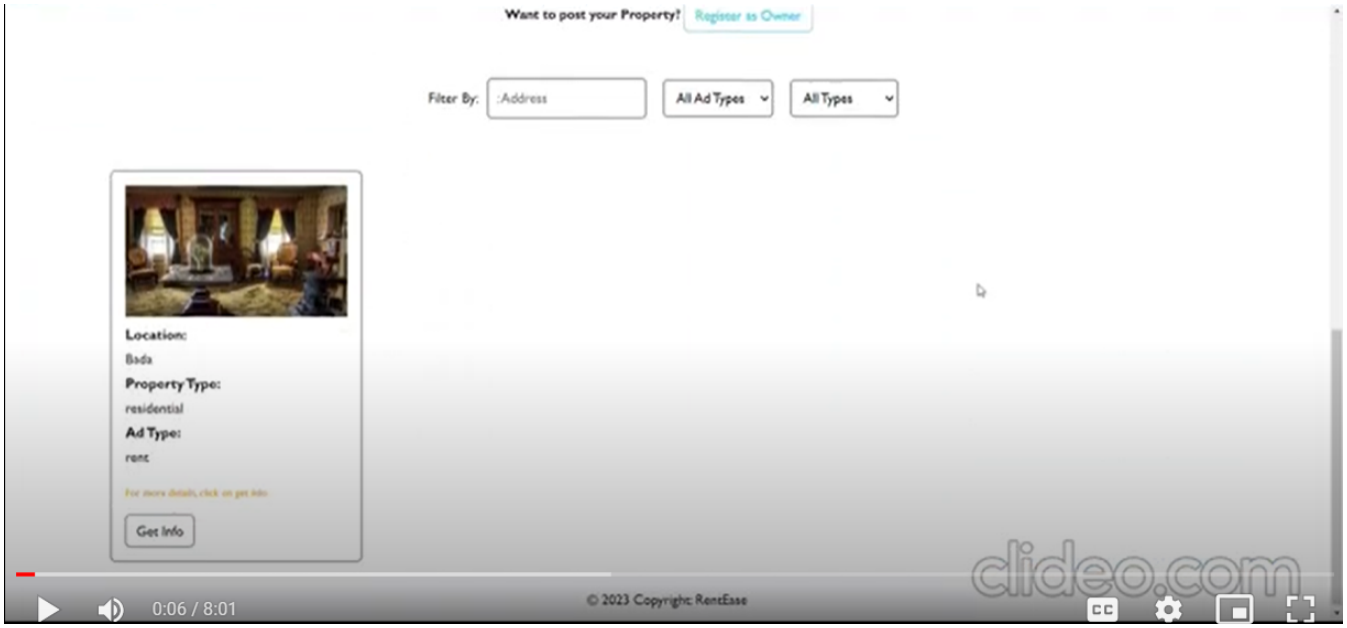
```
const express = require("express");
const dotenv = require("dotenv");
const cors = require("cors");
const connectionofDb = require("./config/connect.js");
const path = require("path");

const app = express();

//////dotenv config////////////////////
dotenv.config();

//////connection to DB/////////////////
connectionofDb();

///////////////port number///////////////////
const PORT = process.env.PORT || 8000;

/////////////////middlewares////////////////
app.use(express.json());
app.use(cors());

/////////////////routes////////////////////
app.use("/uploads", express.static(path.join(__dirname, "uploads")));

app.use('/api/user', require('./routes/userRoutes.js'))
app.use('/api/admin', require('./routes/adminRoutes.js'))
app.use('/api/owner', require('./routes/ownerRoutes.js'))


app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);});
```

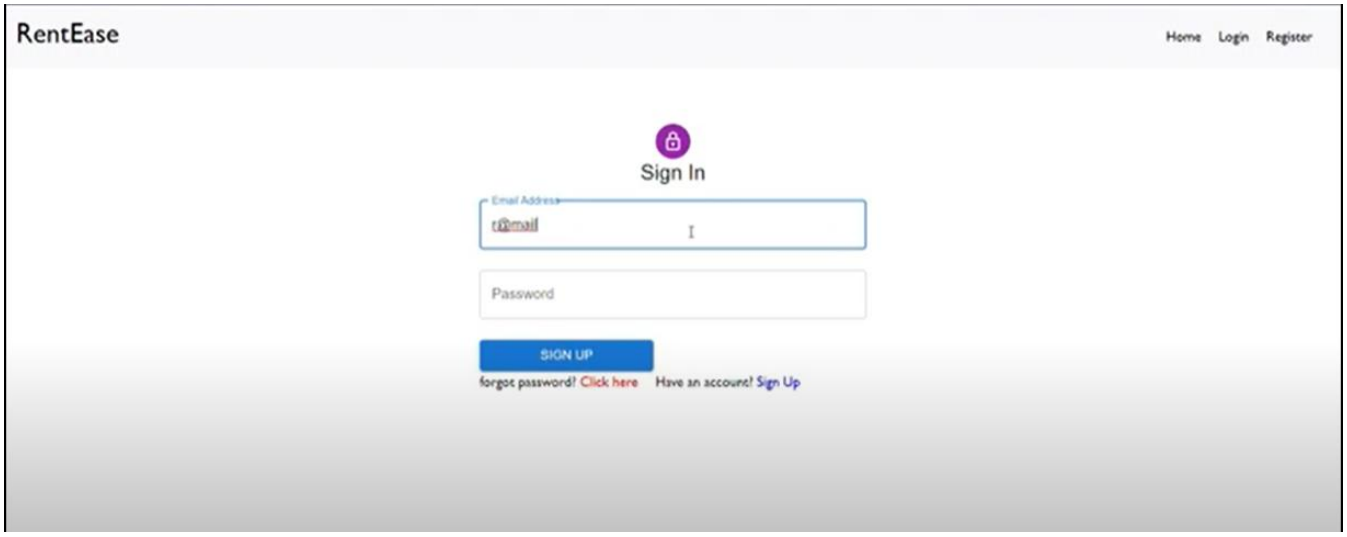
CHAPTER 6

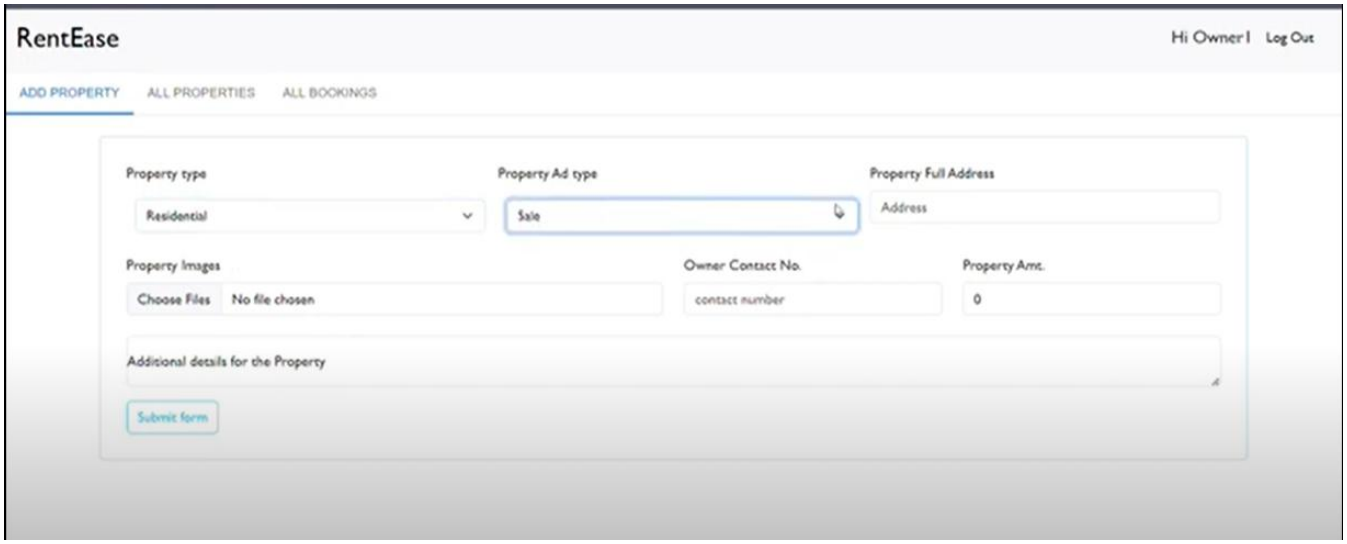
OUTPUT

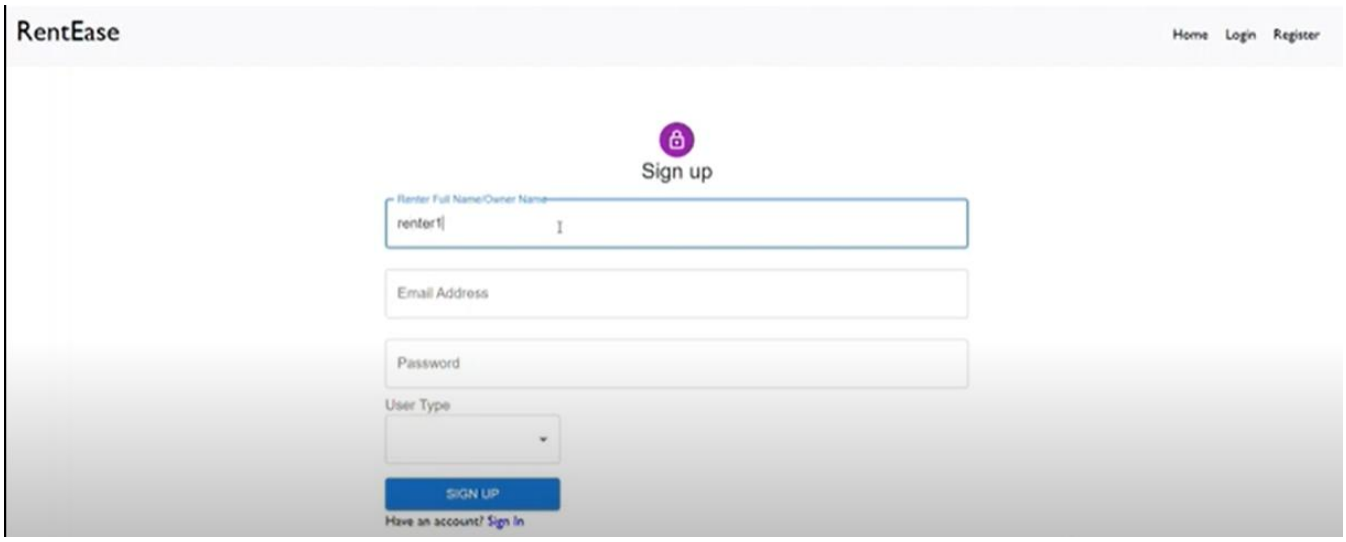


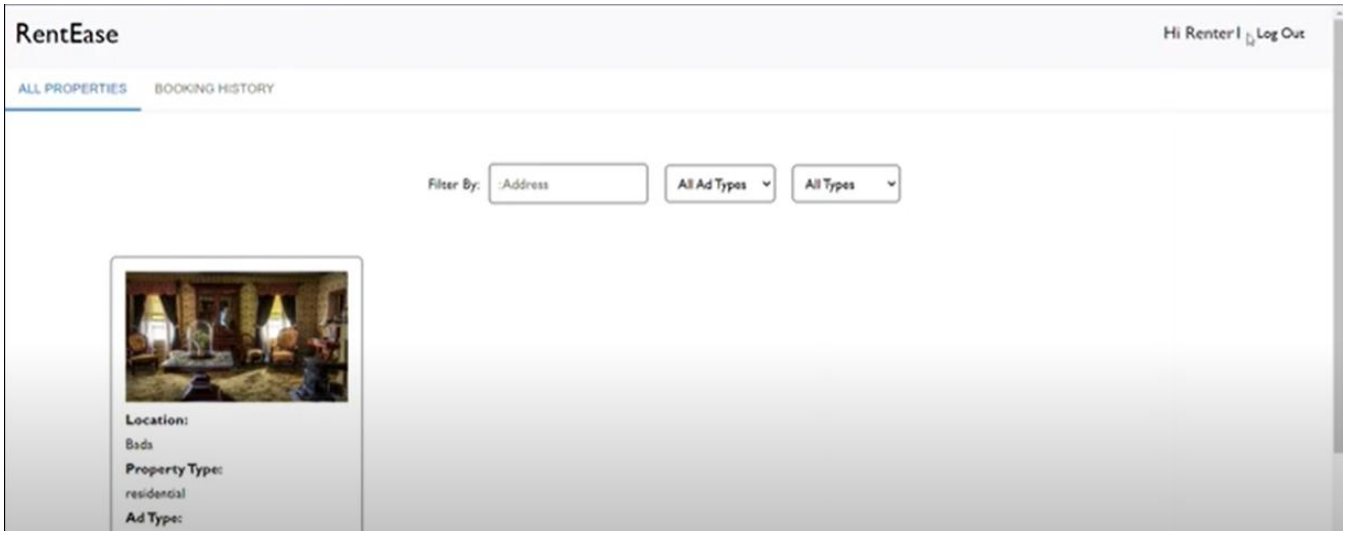



Want to post your Property?
Register as Owner
Filter By:
:Address
All Ad Types
All Types
Location:
Bada
Property Type:
residential
Ad Type:
rent
Get Info
0:06 / 8:01
© 2023 Copyright: RentEase
clideo.com


RentEase
Home
Login
Register
Sign In
Email Address
r@mail
Password
SIGN UP
forgot password? Click here
Have an account? Sign Up


RentEase
Hi Owner!
Log Out
ADD PROPERTY
ALL PROPERTIES
ALL BOOKINGS
Property type
Residential
Property Ad type
Sale
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


RentEase
Home
Login
Register
Sign up
Renter Full Name/Owner Name
renter1
Email Address
Password
User Type
SIGN UP
Have an account? Sign In


RentEase
Hi Renter1
Log Out
ALL PROPERTIES
BOOKING HISTORY
Filter By:
Address
All Ad Types
All Types
Location:
Bada
Property Type:
residential
Ad Type:

CHAPTER 7

CONCLUSION

7.1 Summary

The House Rent App demonstrates the potential of the MERN stack to create a robust, user-friendly solution for the rental market. The platform enhances efficiency, ensures data security, and provides an interactive experience for users.

7.2 FutureWork

Potential future developments include:

- Incorporating AI-based property recommendations.
- Adding voice and chat-based search options.
- Expanding to mobile platforms with React Native.

References

1. Chen & Johnson, Patel & White (2023). Design and Implementation of a House Rental Management System– *International Journal of Novel Research and Development (IJNRD)*,.
2. Ambrose et al. (2021). Online Rental Housing Research Paper. *Journal of Emerging Technologies and Innovative Research (JETIR)*.
3. Wang et al., Garcia et al (2023). Enhancing Transparency in Property Rental Markets through Technology.*International Journal of Novel Research and Development (IJNRD)*.