

1.INTRODUCTION

1.1 Project Description

- CozyStay is a comprehensive hotel management system designed to streamline operations and enhance guest experiences.
- It facilitates online reservations, allowing guests to book, modify, or cancel their stays seamlessly.
- The system maintains detailed guest profiles, capturing preferences and histories to provide personalized services.
- Room management features ensure real-time tracking of availability and status, while billing and invoicing functionalities simplify payment processes.
- With robust reporting and analytics, CozyStay offers valuable insights into occupancy rates and revenue trends.
- User roles and permissions enhance data security, enabling staff to perform their tasks efficiently.
- Additionally, a customer feedback system helps hotels continually improve their offerings. Built with a modern technology stack, CozyStay is mobile-friendly, making it accessible for both guests and staff.
- Overall, CozyStay aims to optimize hotel management processes, increase direct bookings, and elevate the guest experience through tailored services.

1.2 Project Profile

Project Title	Hotel Management System(CozyStay)
Front End	React JS, HTML, JavaScript, CSS, Bootstrap
Back End	Node JS
Database	MongoDB
Team Members	2 Members
Project Duration	2/5 Months
Type Of Application	Web-Application
Platform	Visual studio Code
Operating System	Windows 10
Internal Guide	Prof . Pooja Soni
Submitted By	Jariwala Prachi R
Submitted To	Shree Uttar Gujrat BBA & BCA Collage

2. ENVIRONMENT DESCRIPTION

2.1 Hardware And Software Requirements

Hardware and Software configuration for the implementation environment is as follows.

- **Software Requirements**

Operating System	Microsoft Windows 10
Frontend	VsCode
Backend	Node JS
Scripting Language	JavaScript
DataBase	MongoDB

- **Hardware Requirements**

Processor	Intel Core i5
RAM	8 GB
Hard Disk	238 GB
Disk Space	10 GB Minimum

2.2 Technologies Used

Frontend	VsCode
Backend	Node JS
Scripting Language	JavaScript
DataBase	MongoDB

3. SYSTEM ANALYSIS

3.1 Existing System And It's Drawbacks

- **Existing System**

- When admin can login in the system, admin can able to add rooms. Such as featured rooms, add rooms and also add new arrivals for it. In simple, admin can add and modify the operations.
- Once the user logged in to the system the user cannot able to modify the rooms and the payment gateway is also not available but the user can able to booked the rooms.
- Feedback is also available.

- **Drawbacks**

- When user logged in he/she cannot able to modify the login details and has no permission to modify the room details.
- Admin cannot change his/her profile.
- Customizations options is not available.

3.2 Expected Advantages

- One of the primary benefits is the streamlining of operations; by automating key processes like bookings, billing, and room management, the system significantly reduces manual workloads and minimizes errors.
- This efficiency extends to both staff and guests, thanks to an intuitive interface that enhances user experience.
- Real-time updates are another crucial feature, allowing guests to access up-to-date room availability and make bookings instantly, thereby reducing the risk of overbookings.
- Finally, CozyStay is designed to be cost-efficient.

4. PROPOSED SYSTEM

4.1 Scope

The scope of the system is defined on the basis of Various Functionalities provided by the system. The scope can be explained as :-

- CozyStay hotel management system encompasses a wide range of functionalities designed to enhance both operational efficiency and guest satisfaction.
- First and foremost, the system will provide a robust booking management module that allows guests to book or cancel bookings online, ensuring real-time availability updates.
- Additionally, CozyStay will include a guest profile management feature, enabling hotels to store and analyse detailed information about their guests.
- The scope of the project is to develop customize software package for reducing the manual problems.

4.2 Project Modules

USERS:

- **Users:** Can visit the website and view the rooms.
- **Registration:** Register module is for the user who visit to our website and enter their details for further login in. User must be registered in CozyStay Website when he/she wants to book their rooms.
- **Login:** Login module is also for the user. He/she can only reach the website when it is logged in. User can only login after the registration is done.
- **Room Management:** In this section, user can see different types of rooms and payment details also.
- **Logout:** In this section, user can log out their profile.

ADMIN:

- **Rooms:** In this section, Admin can add , delete and update the rooms.
- **Bookings:** In this section, Admin can view and manage the customer bookings.
- **Users:** In this section, Admin can view the details of users like userID, Name, Email and IsAdmin or Not.

4.3 Objectives/Functionalities

- Provide a user-friendly interface that allows guests to easily navigate the website, make bookings, and access information about the hotel.
- Encourage guests to book directly through the website by offering competitive pricing and exclusive deals.
- Providing various types of rooms such as delux, executive, suite.
- User can easily book their orders by logging in or else they can easily cancel their booking.
- Section for users to leave reviews and feedback.
- Also providing navigation map for location.

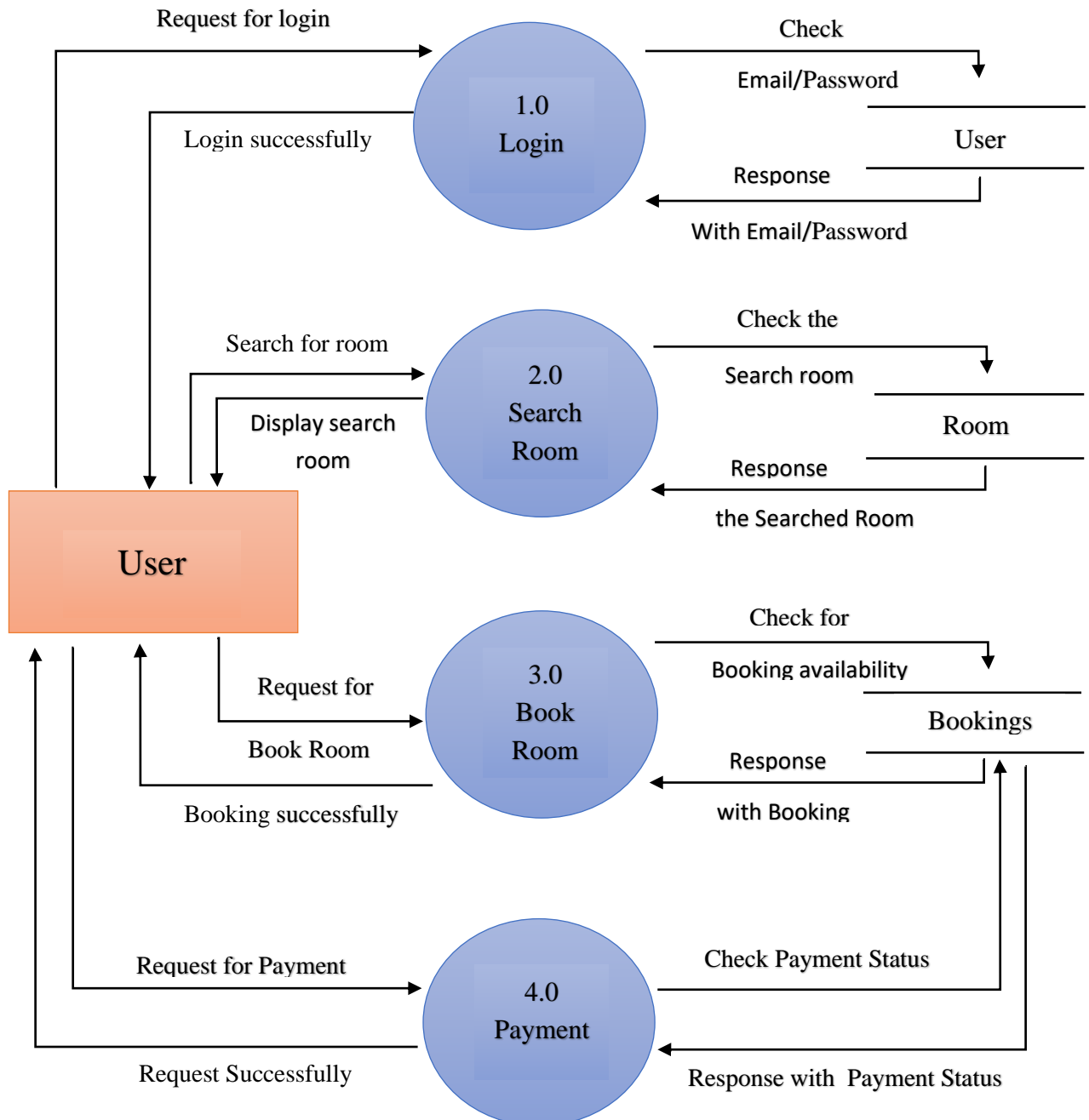
5.DETAIL PLANNING

5.1 Data Flow Diagram

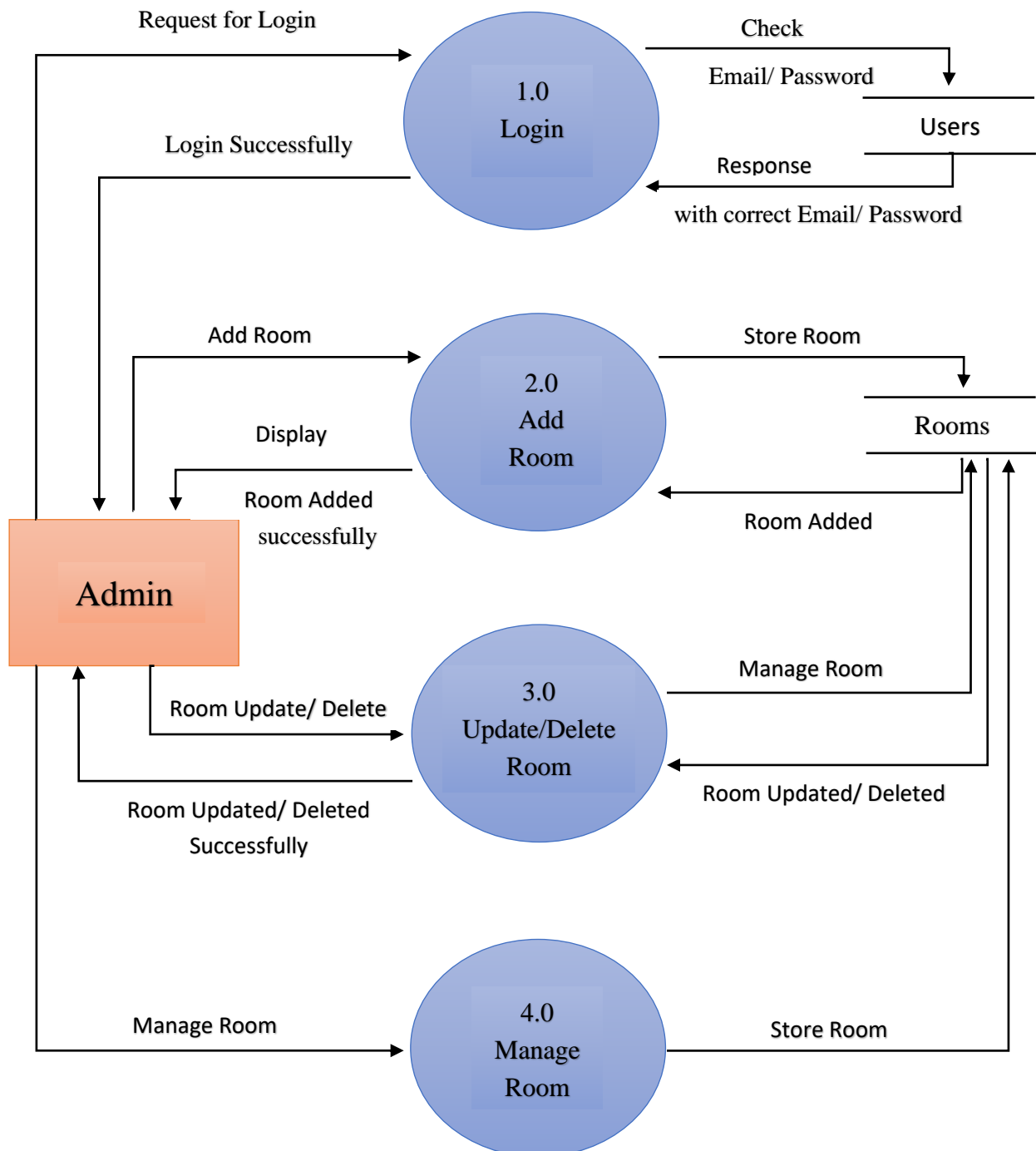
➤ 0 level – Context level



➤ 1 level – User



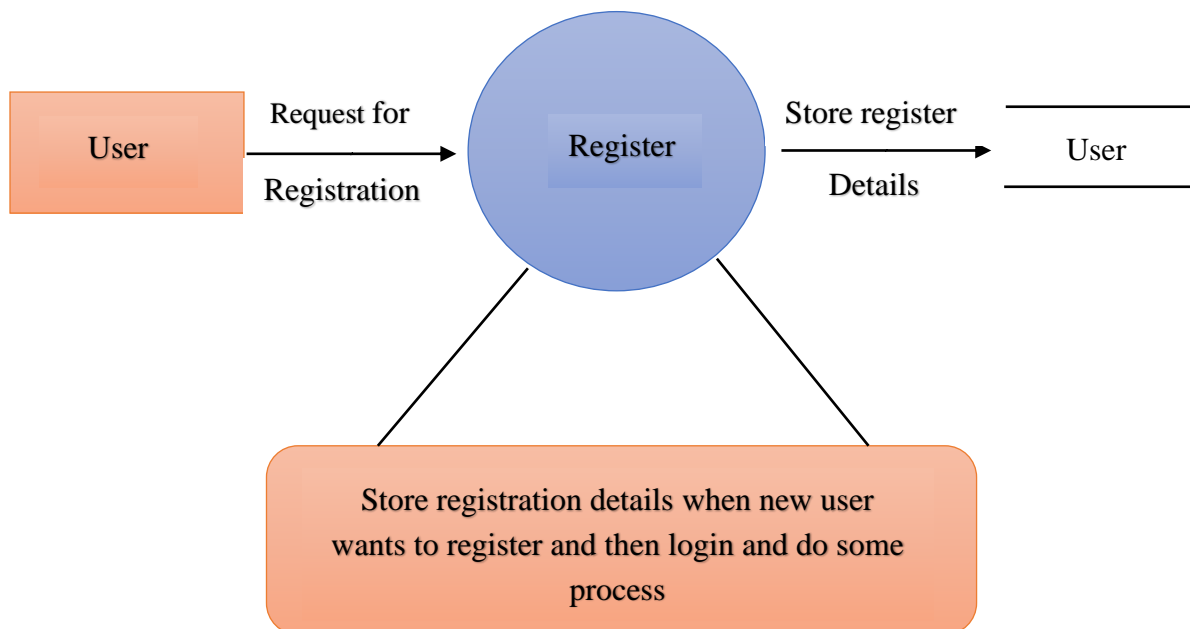
➤ 1 level – Admin



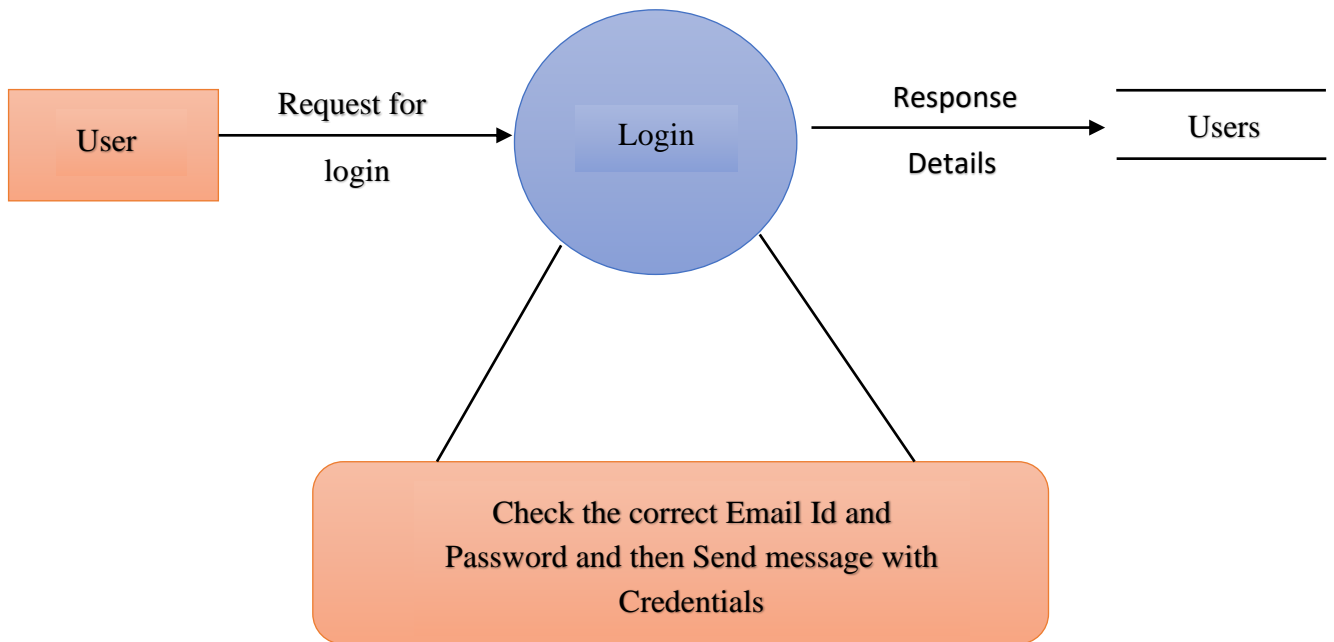
5.2 Process Specification

- A process specification is a method used to document , Analyze and explain the decision-making logic and formulas used to create output making logic and formulas used to create output data from process input data.

➤ Register Process



➤ Login Process



5.3 Data Dictionary

1. Table Name : Users

FIELD_NAME	DATATYPE	SIZE	CONSTRAINTS
ID	Integer	255	Primary key
NAME	String	255	Not null
EMAIL	String	255	Not null
PASSWORD	Integer	255	Not null
IsAdmin	Boolean	-	Check

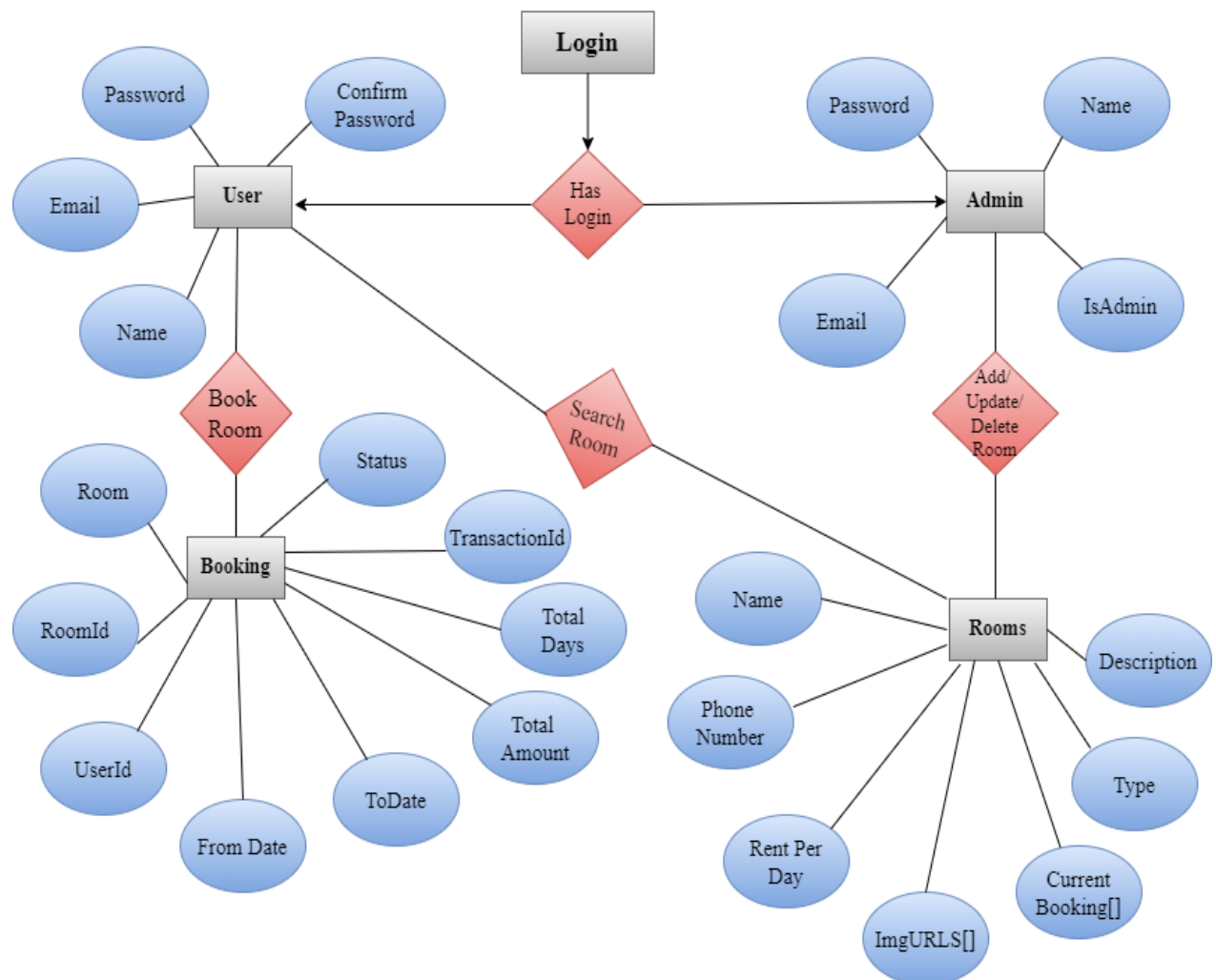
2.Table Name : Rooms

FIELD_NAME	DATATYPE	SIZE	CONSTRAINTS
ID	String	255	Primary key
NAME	String	255	Not null
MAXCOUNT	Number	250	Not null
PHONE NUMBER	Number	250	Not null
RENT PER DAY	Number	250	Not null
IMGURLS[]	String	255	Not null
CURRENT BOOKING	Array	255	-
TYPE	String	255	Not null
DESCRIPTION	String	255	Not null

3.Table Name : Bookings

FIELD_NAME	DATATYPE	SIZE	CONSTRAINTS
ID	String	255	Primary key
ROOM	String	255	Not null
ROOM ID	String	255	Not null
USERID	String	255	Not null
FROM DATE	String	255	Not null
TO DATE	String	255	Not null
TOTAL AMOUNT	Number	255	Not null
TOTAL DAYS	Number	255	Not null
TRANSACTION ID	String	255	Not null
STATUS	String	255	Not null

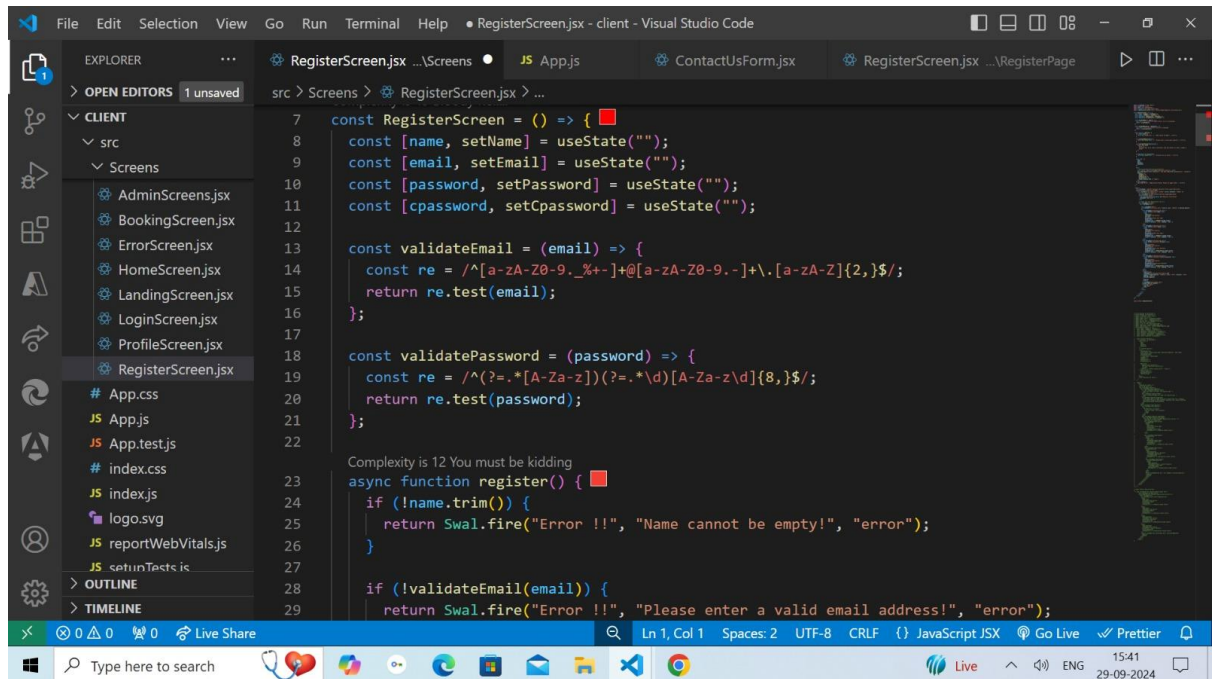
5.4 Entity-Relationship Diagram



6. System Design

6.1 Input Design

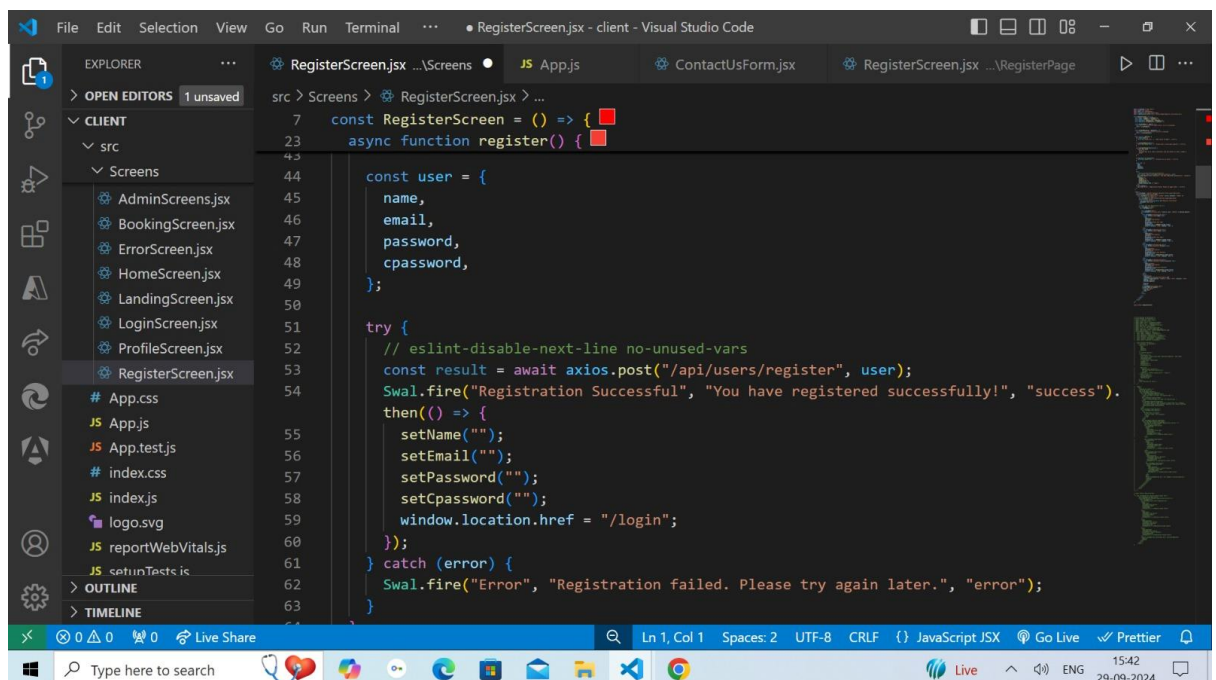
➤ registrationScreen.jsx



```

7  const RegisterScreen = () => {
8    const [name, setName] = useState("");
9    const [email, setEmail] = useState("");
10   const [password, setPassword] = useState("");
11   const [cpassword, setCpassword] = useState("");
12
13   const validateEmail = (email) => {
14     const re = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
15     return re.test(email);
16   };
17
18   const validatePassword = (password) => {
19     const re = /^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/;
20     return re.test(password);
21   };
22
23   Complexity is 12 You must be kidding
24   async function register() {
25     if (!name.trim()) {
26       return Swal.fire("Error !!", "Name cannot be empty!", "error");
27     }
28     if (!validateEmail(email)) {
29       return Swal.fire("Error !!", "Please enter a valid email address!", "error");

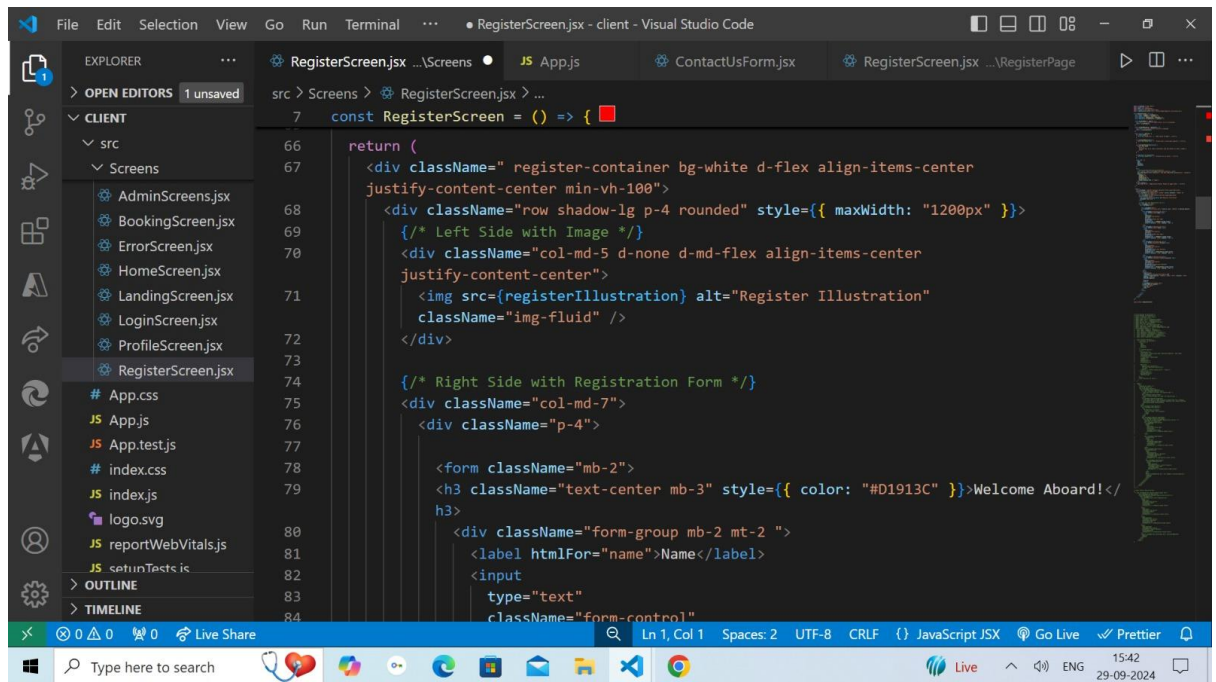
```



```

43   async function register() {
44     const user = {
45       name,
46       email,
47       password,
48       cpassword,
49     };
50
51     try {
52       // eslint-disable-next-line no-unused-vars
53       const result = await axios.post("/api/users/register", user);
54       Swal.fire("Registration Successful", "You have registered successfully!", "success").
55       then(() => {
56         setName("");
57         setEmail("");
58         setPassword("");
59         setCpassword("");
60         window.location.href = "/login";
61       });
62     } catch (error) {
63       Swal.fire("Error", "Registration failed. Please try again later.", "error");
64     }
65   }

```

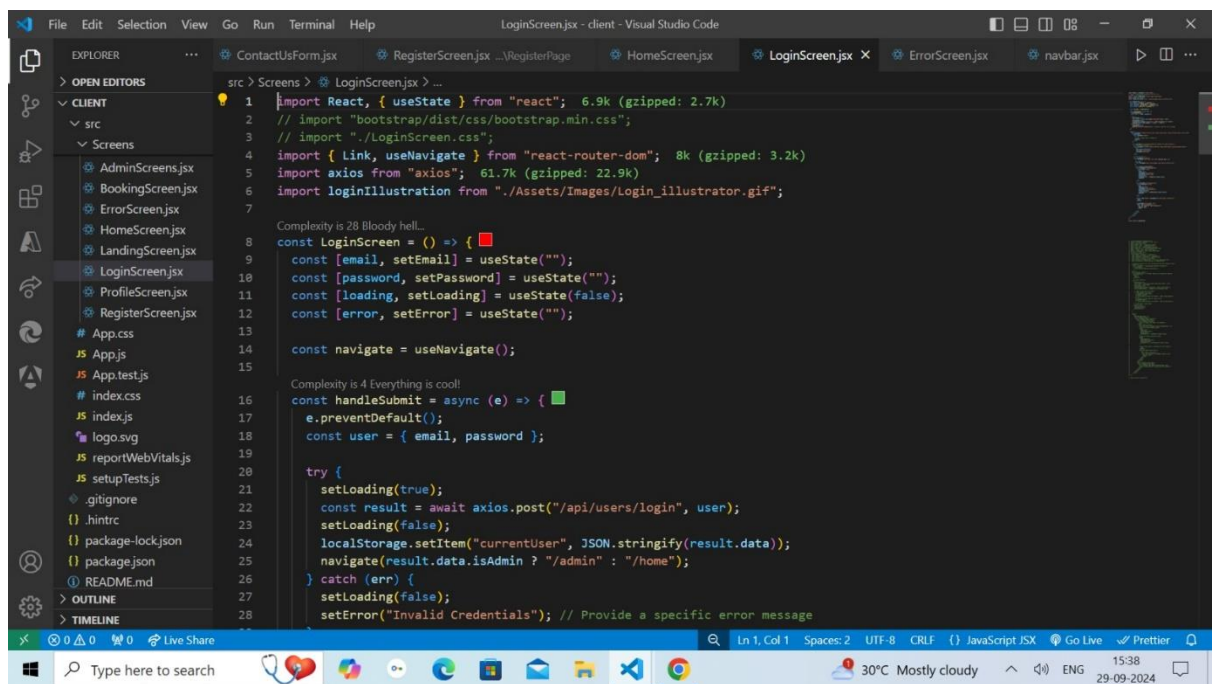


```

7  const RegisterScreen = () => {
66  return (
67    <div className="register-container bg-white d-flex align-items-center
        justify-content-center min-vh-100">
68      <div className="row shadow-lg p-4 rounded" style={{ maxWidth: "1200px" }}>
69        /* Left Side with Image */
70        <div className="col-md-5 d-none d-md-flex align-items-center
        justify-content-center">
71          <img src={registerIllustration} alt="Register Illustration"
72            className="img-fluid" />
73        </div>
74        /* Right Side with Registration Form */
75        <div className="col-md-7">
76          <div className="p-4">
77            <form className="mb-2">
78              <h3 className="text-center mb-3" style={{ color: "#D1913C" }}>Welcome Aboard!</
79              h3>
80              <div className="form-group mb-2 mt-2 ">
81                <label htmlFor="name">Name</label>
82                <input
83                  type="text"
84                  className="form-control"

```

➤ loginScreen.jsx



```

1  import React, { useState } from "react"; 6.9k (gzipped: 2.7k)
2  // import "bootstrap/dist/css/bootstrap.min.css";
3  // import "../LoginScreen.css";
4  import { Link, useNavigate } from "react-router-dom"; 8k (gzipped: 3.2k)
5  import axios from "axios"; 61.7k (gzipped: 22.9k)
6  import loginIllustration from "../Assets/Images/Login_illustrator.gif";
7
8  const LoginScreen = () => {
9    const [email, setEmail] = useState("");
10   const [password, setPassword] = useState("");
11   const [loading, setLoading] = useState(false);
12   const [error, setError] = useState("");
13
14   const navigate = useNavigate();
15
16   const handleSubmit = async (e) => {
17     e.preventDefault();
18     const user = { email, password };
19
20     try {
21       setLoading(true);
22       const result = await axios.post("/api/users/login", user);
23       setLoading(false);
24       localStorage.setItem("currentUser", JSON.stringify(result.data));
25       navigate(result.data.isAdmin ? "/admin" : "/home");
26     } catch (err) {
27       setLoading(false);
28       setError("Invalid Credentials"); // Provide a specific error message

```



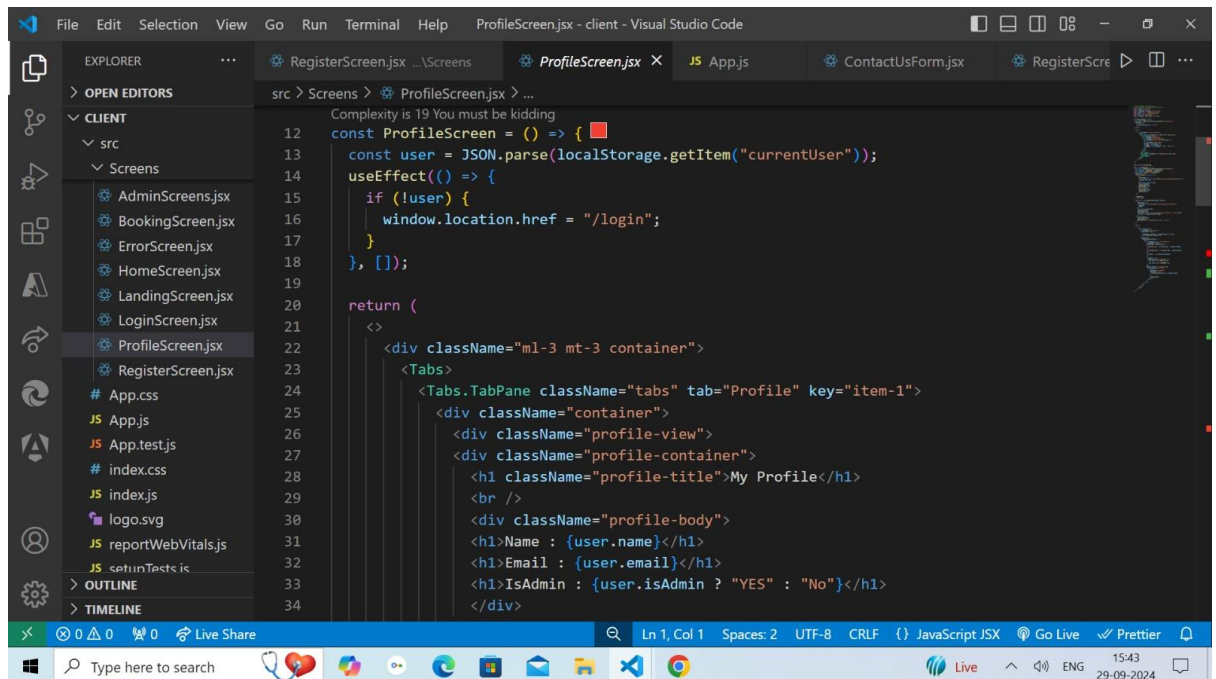
```
File Edit Selection View Go Run Terminal Help LoginScreen.jsx - client - Visual Studio Code

EXPLORER
src > Screens > LoginScreen.jsx > ...
8 const LoginScreen = () => {
16 const handleSubmit = async (e) => {
26 } catch (err) {
27   setloading(false);
28   setError("Invalid Credentials"); // Provide a specific error message
29 }
30 };
31
32 return (
33   <div className="container-fluid d-flex align-items-center justify-content-center min-vh-100">
34     {loading && (
35       <div className="loader">
36         {" "}
37         {/* Add a loader here if needed */}
38         Loading...
39       </div>
40     )}
41
42     <div className="row bg-white shadow-lg rounded p-4 login-container">
43       {/* Illustration Section */}
44       <div className="col-md-6 d-none d-md-flex align-items-center justify-content-center">
45         <img
```

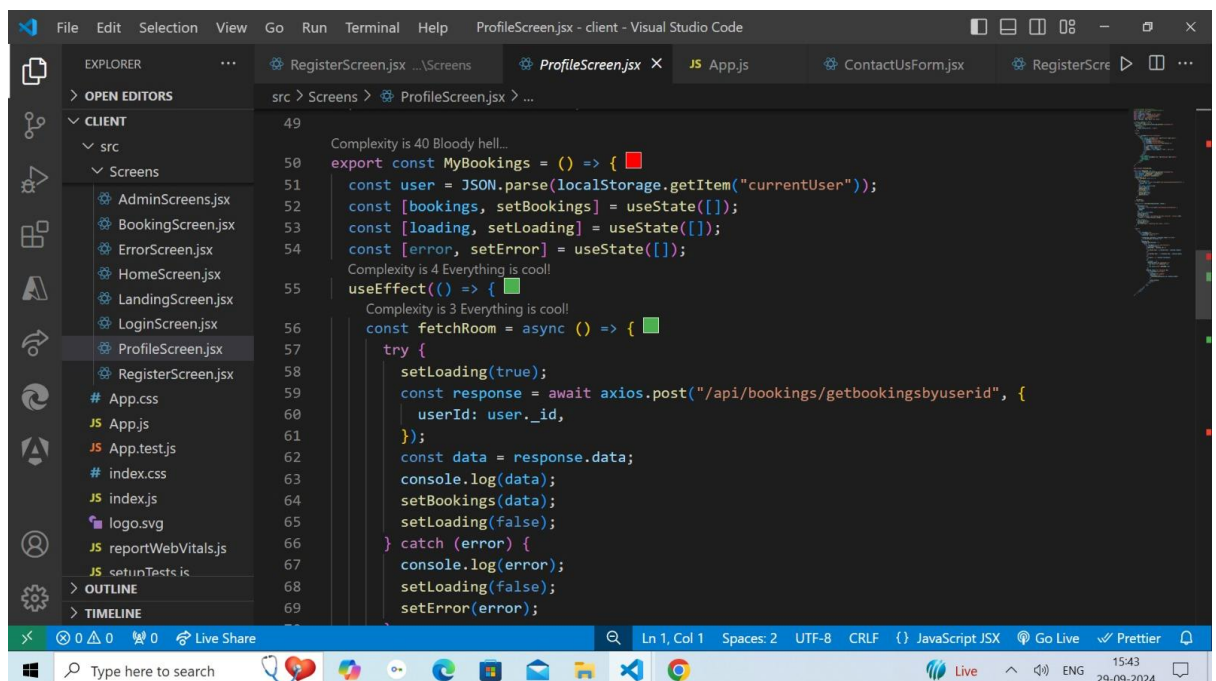
```
File Edit Selection View Go Run Terminal Help LoginScreen.jsx - client - Visual Studio Code

EXPLORER
src > Screens > LoginScreen.jsx > ...
8 const LoginScreen = () => {
32 return (
33   <div className="container-fluid d-flex align-items-center justify-content-center min-vh-100">
34     {loading && (
35       <div className="loader">
36         {" "}
37         {/* Add a loader here if needed */}
38         Loading...
39       </div>
40     )}
41
42     <div className="row bg-white shadow-lg rounded p-4 login-container">
43       {/* Illustration Section */}
44       <div className="col-md-6 d-none d-md-flex align-items-center justify-content-center">
45         <img
46           src={loginIllustration}
47           alt="Login Illustration"
48           className="img-fluid"
49         />
50       </div>
51
52       {/* Form Section */}
53       <div className="col-md-6">
54         <h3 className="text-center mt-2 mb-0">Welcome Back!</h3>
55
56         {error && (
57           <div className="alert alert-danger text-center">{error}</div>
58         )}
59
60         <form onSubmit={handleSubmit}>
```

➤ profileScreen.jsx

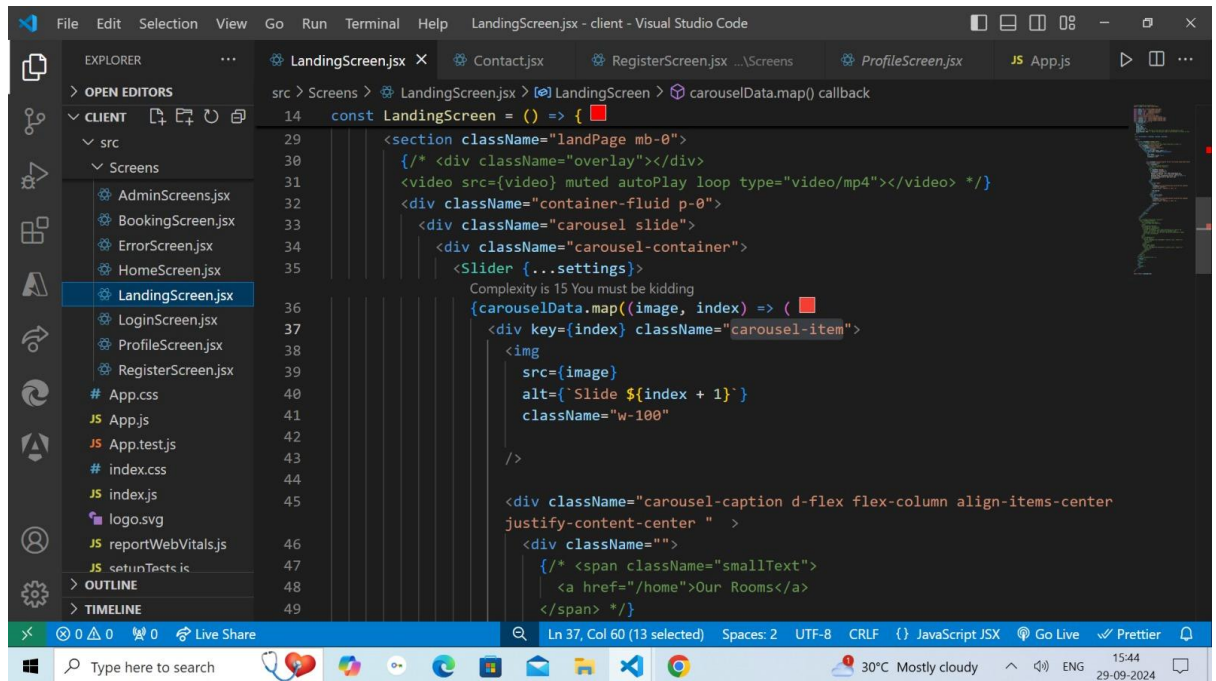


```
12  const ProfileScreen = () => {  
13    const user = JSON.parse(localStorage.getItem("currentUser"));  
14    useEffect(() => {  
15      if (!user) {  
16        window.location.href = "/login";  
17      }  
18    }, []);  
19  
20    return (  
21      <>  
22        <div className="ml-3 mt-3 container">  
23          <Tabs>  
24            <Tabs.TabPane className="tabs" tab="Profile" key="item-1">  
25              <div className="container">  
26                <div className="profile-view">  
27                  <div className="profile-container">  
28                    <h1 className="profile-title">My Profile</h1>  
29                    <br />  
30                    <div className="profile-body">  
31                      <h1>Name : {user.name}</h1>  
32                      <h1>Email : {user.email}</h1>  
33                      <h1>IsAdmin : {user.isAdmin ? "YES" : "No"}</h1>  
34                    </div>  
35                  </div>  
36                </div>  
37              </div>  
38            </Tabs>  
39          </div>  
40        </div>  
41      </>  
42    );  
43  }  
44
```



```
49  export const MyBookings = () => {  
50    const user = JSON.parse(localStorage.getItem("currentUser"));  
51    const [bookings, setBookings] = useState([]);  
52    const [loading, setLoading] = useState(false);  
53    const [error, setError] = useState("");  
54    useEffect(() => {  
55      const fetchRoom = async () => {  
56        try {  
57          setLoading(true);  
58          const response = await axios.post("/api/bookings/getbookingsbyuserid", {  
59            userId: user._id,  
60          });  
61          const data = response.data;  
62          console.log(data);  
63          setBookings(data);  
64          setLoading(false);  
65        } catch (error) {  
66          console.log(error);  
67          setError(error);  
68        }  
69      }  
70    }, [user._id]);  
71  }  
72
```

➤ landingScreen.jsx

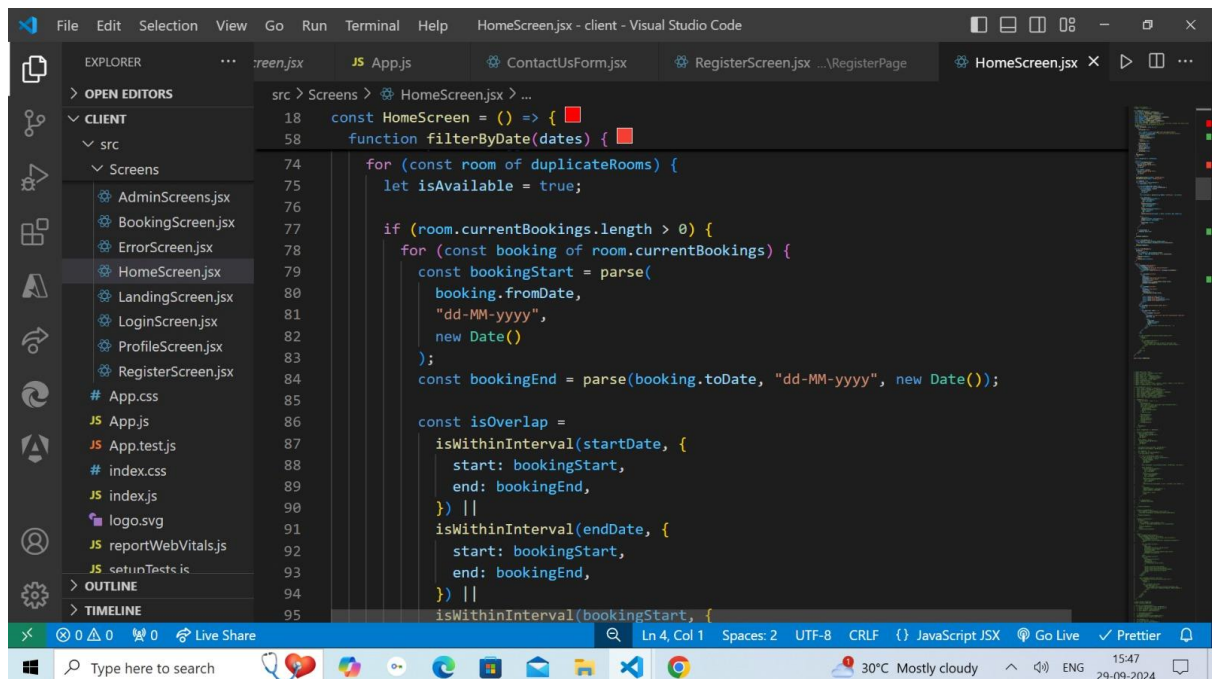


```

14  const LandingScreen = () => {
15    <section className="landPage mb-0">
16      <div className="overlay"></div>
17      <video src={video} muted autoPlay loop type="video/mp4"></video> </>
18      <div className="container-fluid p-0">
19        <div className="carousel slide">
20          <div className="carousel-container">
21            <Slider {...settings}>
22              <div key={index} className="carousel-item">
23                <img
24                  src={image}
25                  alt={`Slide ${index + 1}`}
26                  className="w-100"
27                />
28              <div className="carousel-caption d-flex flex-column align-items-center justify-content-center">
29                <div className="">
30                  <span className="smallText">
31                    <a href="/home">Our Rooms</a>
32                  </span> </>
33                </div>
34              </div>
35            </div>
36          </div>
37        </div>
38      </div>
39    </section>
40  </div>
41  </div>
42  </div>
43  </div>
44  </div>
45  </div>
46  </div>
47  </div>
48  </div>
49  </div>

```

➤ homeScreen.jsx

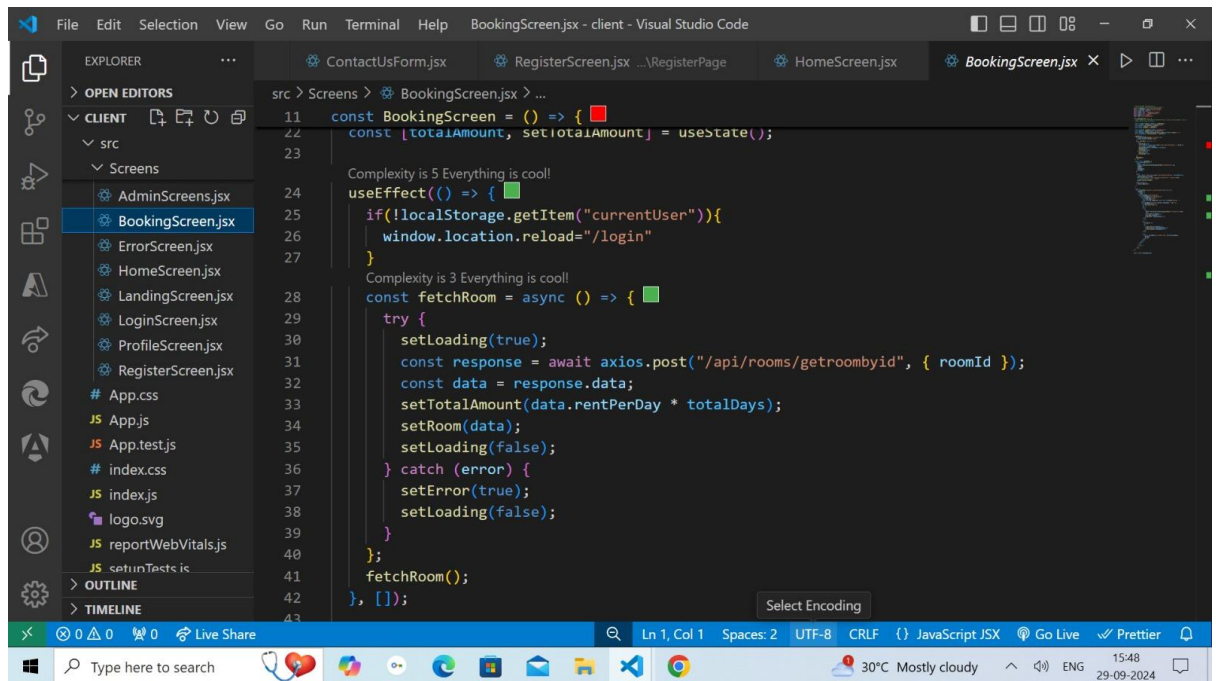


```

18  const HomeScreen = () => {
19    function filterByDate(dates) {
20      for (const room of duplicateRooms) {
21        let isAvailable = true;
22        if (room.currentBookings.length > 0) {
23          for (const booking of room.currentBookings) {
24            const bookingStart = parse(
25              booking.fromDate,
26              "dd-MM-yyyy",
27              new Date()
28            );
29            const bookingEnd = parse(booking.toDate, "dd-MM-yyyy", new Date());
30            const isOverlap =
31              isWithinInterval(startDate, {
32                start: bookingStart,
33                end: bookingEnd,
34              }) ||
35              isWithinInterval(endDate, {
36                start: bookingStart,
37                end: bookingEnd,
38              }) ||
39              isWithinInterval(bookingStart, {
40                start: bookingStart,
41                end: bookingEnd,
42              })
43            if (isOverlap) {
44              isAvailable = false;
45            }
46          }
47        }
48        if (isAvailable) {
49          rooms.push(room);
50        }
51      }
52    }
53  }
54  </div>
55  </div>
56  </div>
57  </div>
58  </div>
59  </div>
60  </div>
61  </div>
62  </div>
63  </div>
64  </div>
65  </div>
66  </div>
67  </div>
68  </div>
69  </div>
70  </div>
71  </div>
72  </div>
73  </div>
74  </div>
75  </div>
76  </div>
77  </div>
78  </div>
79  </div>
80  </div>
81  </div>
82  </div>
83  </div>
84  </div>
85  </div>
86  </div>
87  </div>
88  </div>
89  </div>
90  </div>
91  </div>
92  </div>
93  </div>
94  </div>
95  </div>

```


➤ BookingScreen.jsx

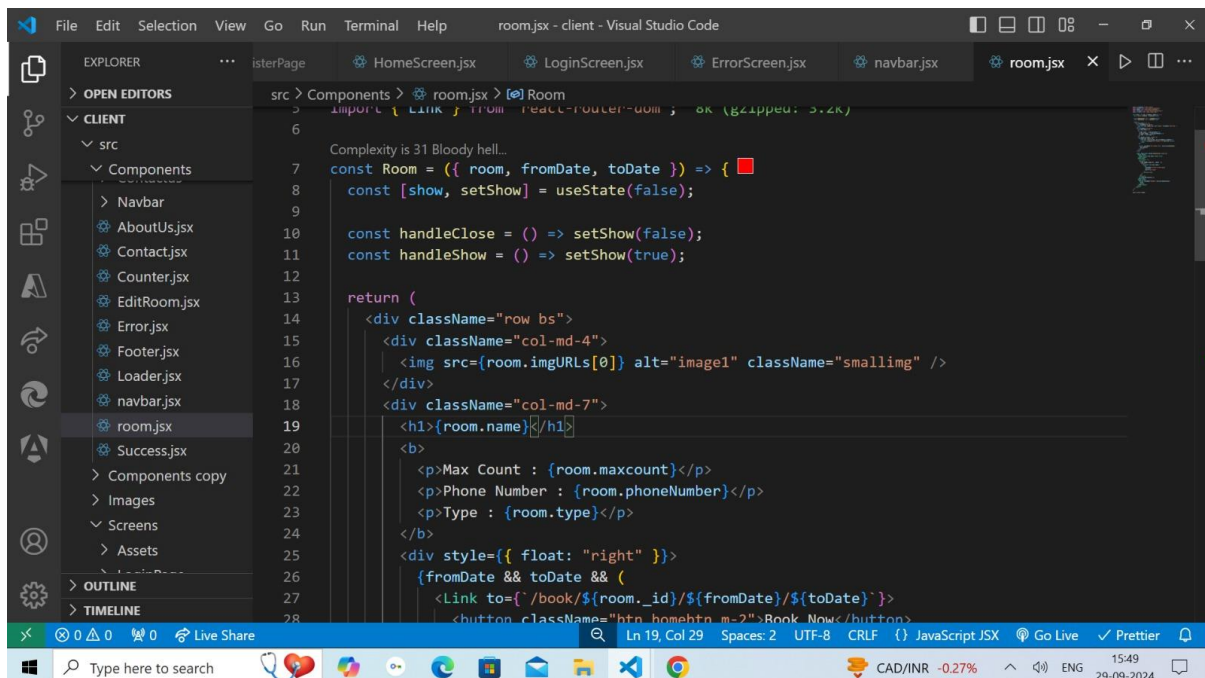


```
File Edit Selection View Go Run Terminal Help BookingScreen.jsx - client - Visual Studio Code

EXPLORER
  > OPEN EDITORS
  > CLIENT
    > src
      > Screens
        BookingScreen.jsx
        AdminScreens.jsx
        ErrorScreen.jsx
        HomeScreen.jsx
        LandingScreen.jsx
        LoginScreen.jsx
        ProfileScreen.jsx
        RegisterScreen.jsx
        App.css
        App.js
        App.test.js
        index.css
        index.js
        logo.svg
        reportWebVitals.js
        setupTests.js
    > OUTLINE
    > TIMELINE

src > Screens > BookingScreen.jsx > ...
11 const BookingScreen = () => {
22   const [totalAmount, setTotalAmount] = useState();
23
24   Complexity is 5 Everything is cool!
25   useEffect(() => {
26     if(!localStorage.getItem("currentUser")){
27       window.location.reload="/login"
28     }
29
30     Complexity is 3 Everything is cool!
31     const fetchRoom = async () => {
32       try {
33         setLoading(true);
34         const response = await axios.post("/api/rooms/getroombyid", { roomId });
35         const data = response.data;
36         setTotalAmount(data.rentPerDay * totalDays);
37         setRoom(data);
38         setLoading(false);
39       } catch (error) {
40         setError(true);
41         setLoading(false);
42       }
43     };
44     fetchRoom();
45   }, []);
46 }
```

➤ room.jsx

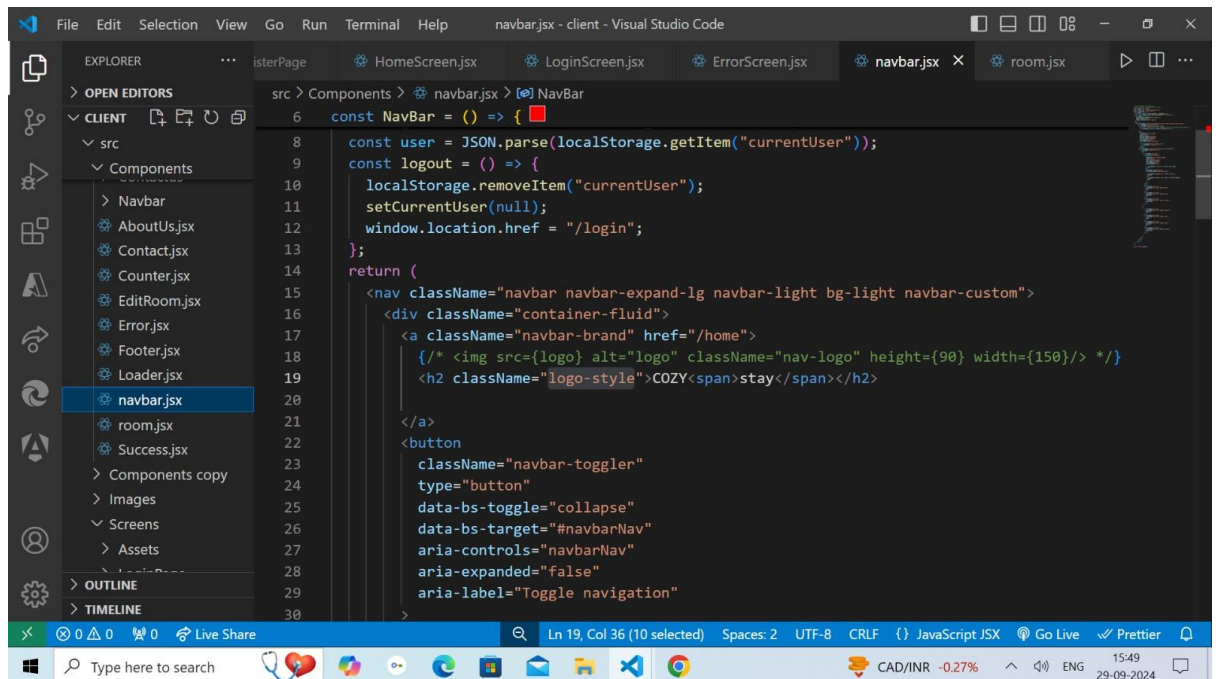


```
File Edit Selection View Go Run Terminal Help room.jsx - client - Visual Studio Code

EXPLORER
  > OPEN EDITORS
  > CLIENT
    > src
      > Components
        room.jsx
        AboutUs.jsx
        Contact.jsx
        Counter.jsx
        EditRoom.jsx
        Error.jsx
        Footer.jsx
        Loader.jsx
        navbar.jsx
        Success.jsx
        Components copy
        Images
        Screens
        Assets
    > OUTLINE
    > TIMELINE

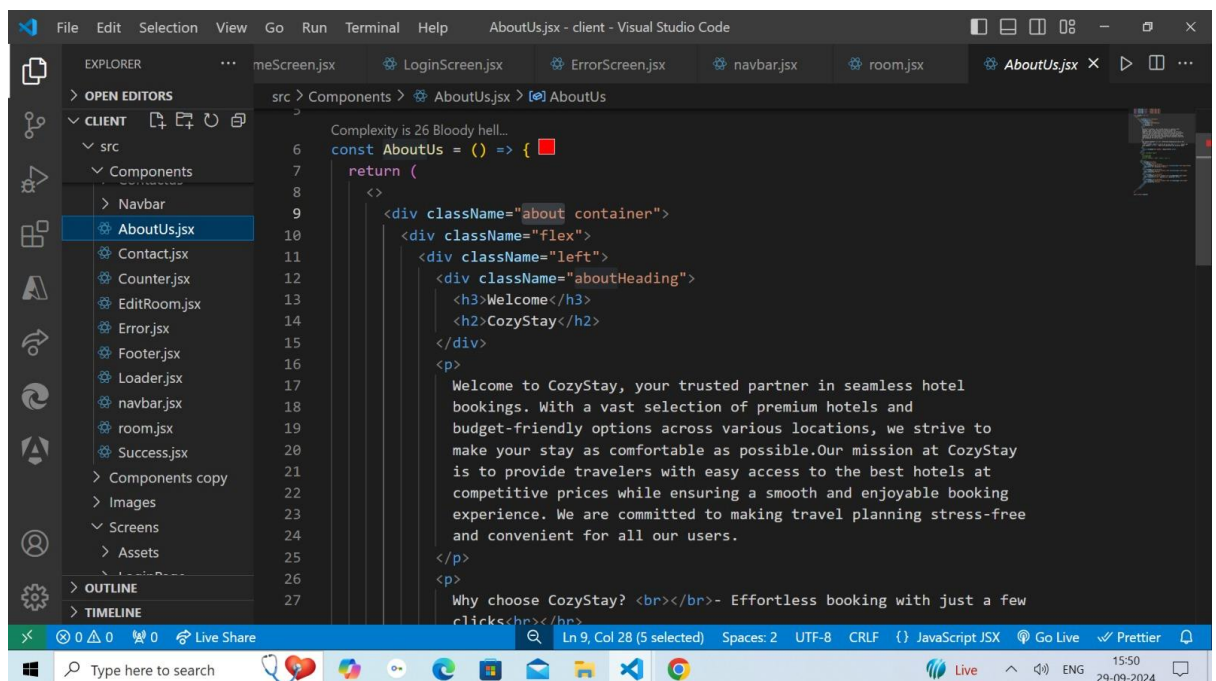
src > Components > room.jsx > Room
6 import { Link } from 'react-router-dom'; // OK (eslint: 3.2K)
7
8 Complexity is 31 Bloody hell...
9 const Room = ({ room, fromDate, toDate }) => {
10   const [show, setShow] = useState(false);
11
12   const handleClose = () => setShow(false);
13   const handleShow = () => setShow(true);
14
15   return (
16     <div className="row bs">
17       <div className="col-md-4">
18         <img src={room.imgURLs[0]} alt="image1" className="smallimg" />
19       </div>
20       <div className="col-md-7">
21         <h1>{room.name}</h1>
22         <b>
23           <p>Max Count : {room.maxcount}</p>
24           <p>Phone Number : {room.phoneNumber}</p>
25           <p>Type : {room.type}</p>
26         </b>
27         <div style={{ float: "right" }}>
28           {fromDate && toDate && (
29             <Link to={`/book/${room.id}/${fromDate}/${toDate}`}>
30               <button className="btn btn-primary">Book Now</button>
31             )
32           }
33         </div>
34       </div>
35     </div>
36   );
37 }
```

➤ navbar.jsx



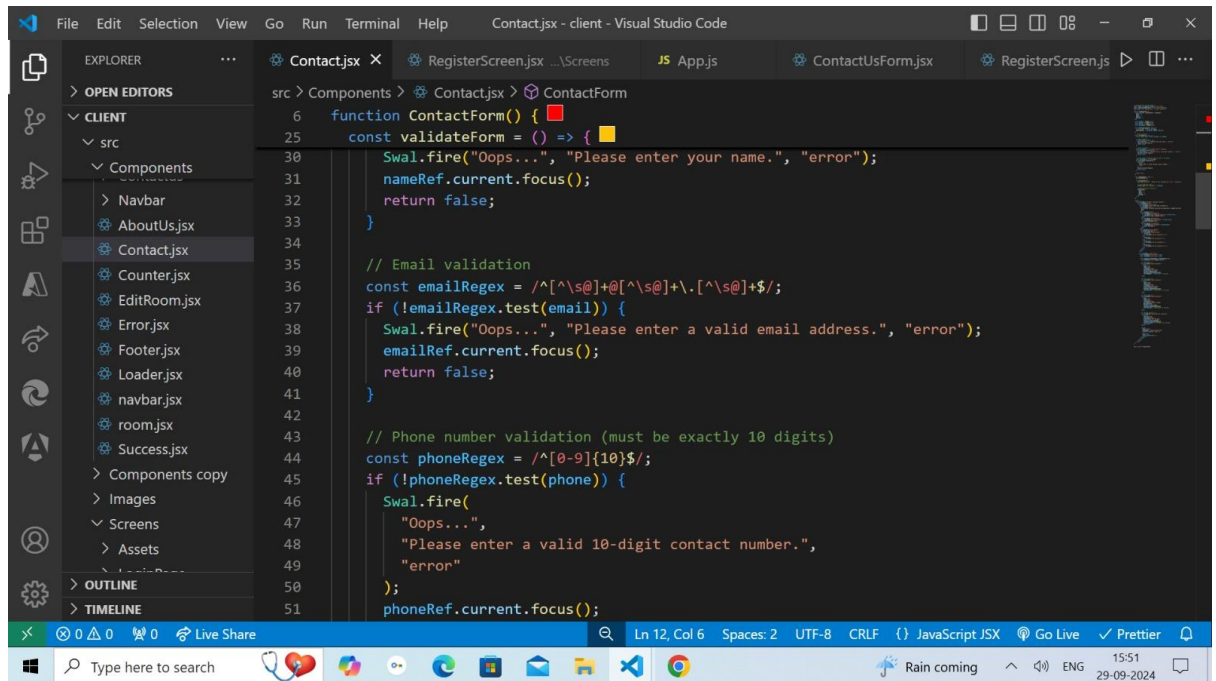
```
6 const NavBar = () => {  
8   const user = JSON.parse(localStorage.getItem("currentUser"));  
9   const logout = () => {  
10     localStorage.removeItem("currentUser");  
11     setCurrentUser(null);  
12     window.location.href = "/login";  
13   };  
14   return (  
15     <nav className="navbar navbar-expand-lg navbar-light bg-light navbar-custom">  
16       <div className="container-fluid">  
17         <a className="navbar-brand" href="/home">  
18           /* <img src={logo} alt="logo" className="nav-logo" height={90} width={150}/> */  
19           <h2 className="logo-style">COZY<span>stay</span></h2>  
20         </a>  
21         <button  
22           className="navbar-toggler"  
23           type="button"  
24           data-bs-toggle="collapse"  
25           data-bs-target="#navbarNav"  
26           aria-controls="navbarNav"  
27           aria-expanded="false"  
28           aria-label="Toggle navigation"  
29         >  
30
```

➤ aboutUs.jsx



```
6 const AboutUs = () => {  
7   return (  
8     <>  
9     <div className="about container">  
10       <div className="flex">  
11         <div className="left">  
12           <div className="aboutHeading">  
13             <h3>Welcome</h3>  
14             <h2>CozyStay</h2>  
15           </div>  
16           <p>  
17             Welcome to CozyStay, your trusted partner in seamless hotel  
18             bookings. With a vast selection of premium hotels and  
19             budget-friendly options across various locations, we strive to  
20             make your stay as comfortable as possible. Our mission at CozyStay  
21             is to provide travelers with easy access to the best hotels at  
22             competitive prices while ensuring a smooth and enjoyable booking  
23             experience. We are committed to making travel planning stress-free  
24             and convenient for all our users.  
25           </p>  
26           <p>  
27             Why choose CozyStay? <br><br>- Effortless booking with just a few  
28             clicks<br><br>  
29           </p>  
30
```

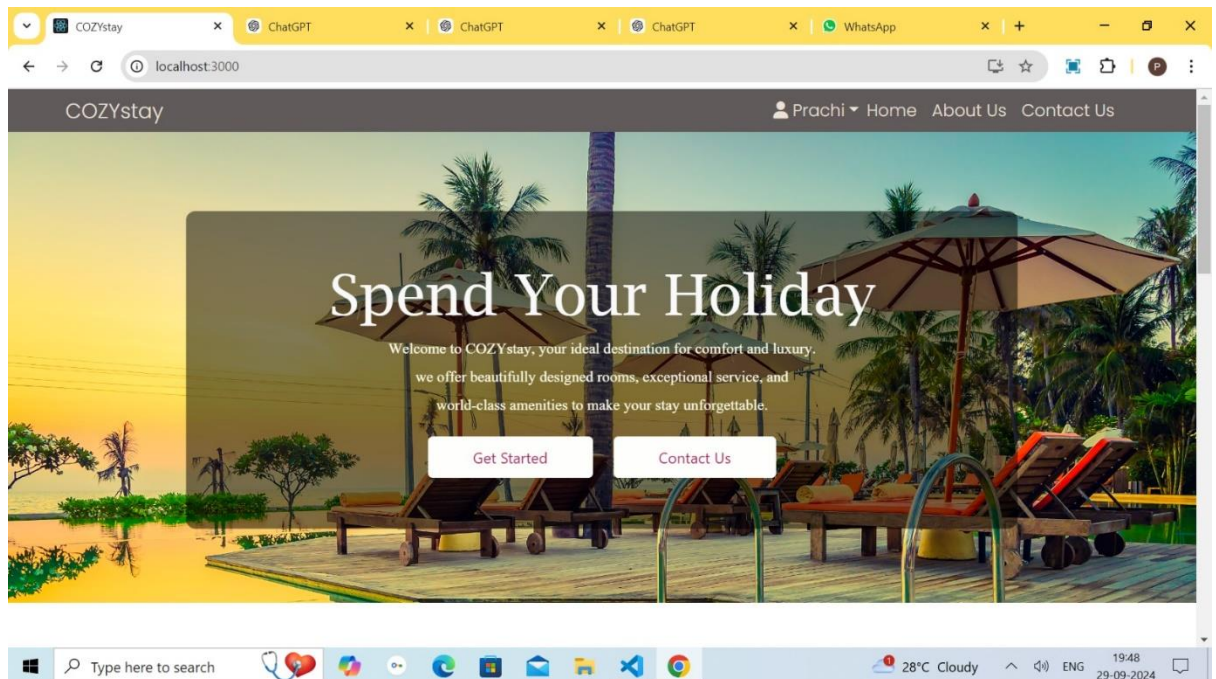
➤ contact.jsx



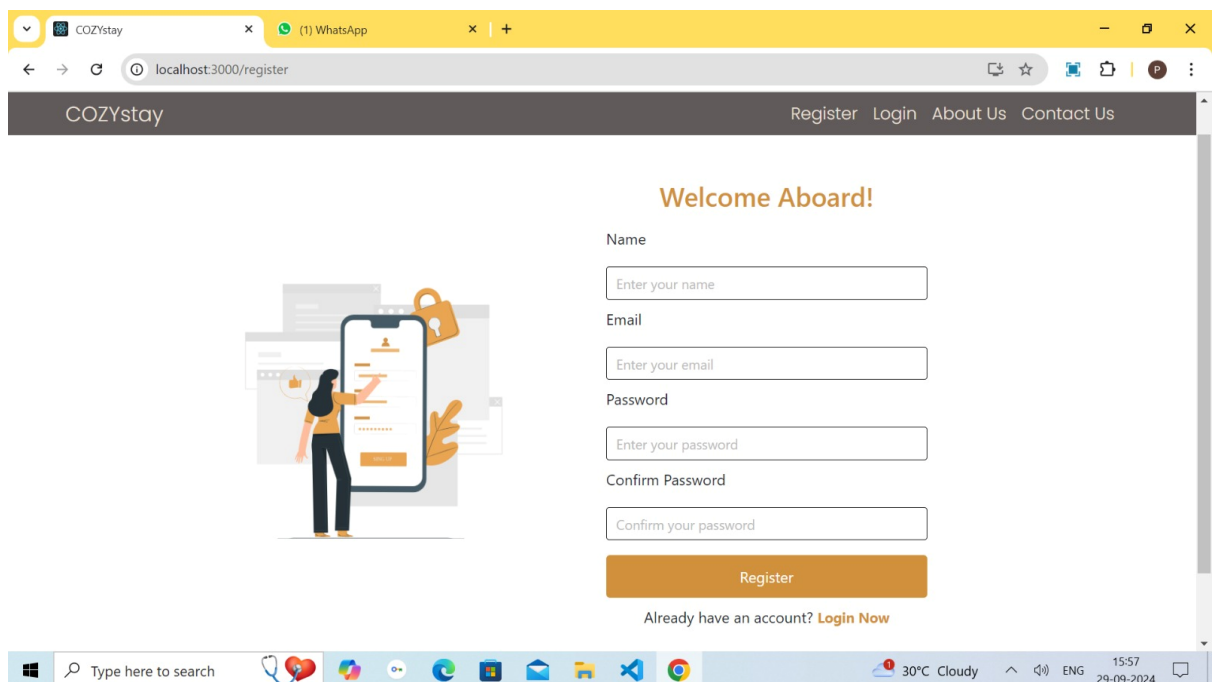
```
6 function ContactForm() {
25   const validateForm = () => {
30     Swal.fire("Oops...", "Please enter your name.", "error");
31     nameRef.current.focus();
32     return false;
33   }
34
35   // Email validation
36   const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
37   if (!emailRegex.test(email)) {
38     Swal.fire("Oops...", "Please enter a valid email address.", "error");
39     emailRef.current.focus();
40     return false;
41   }
42
43   // Phone number validation (must be exactly 10 digits)
44   const phoneRegex = /^[0-9]{10}$/;
45   if (!phoneRegex.test(phone)) {
46     Swal.fire(
47       "Oops...",
48       "Please enter a valid 10-digit contact number.",
49       "error"
50     );
51     phoneRef.current.focus();
52   }
53 }
```

6.2 Output Design

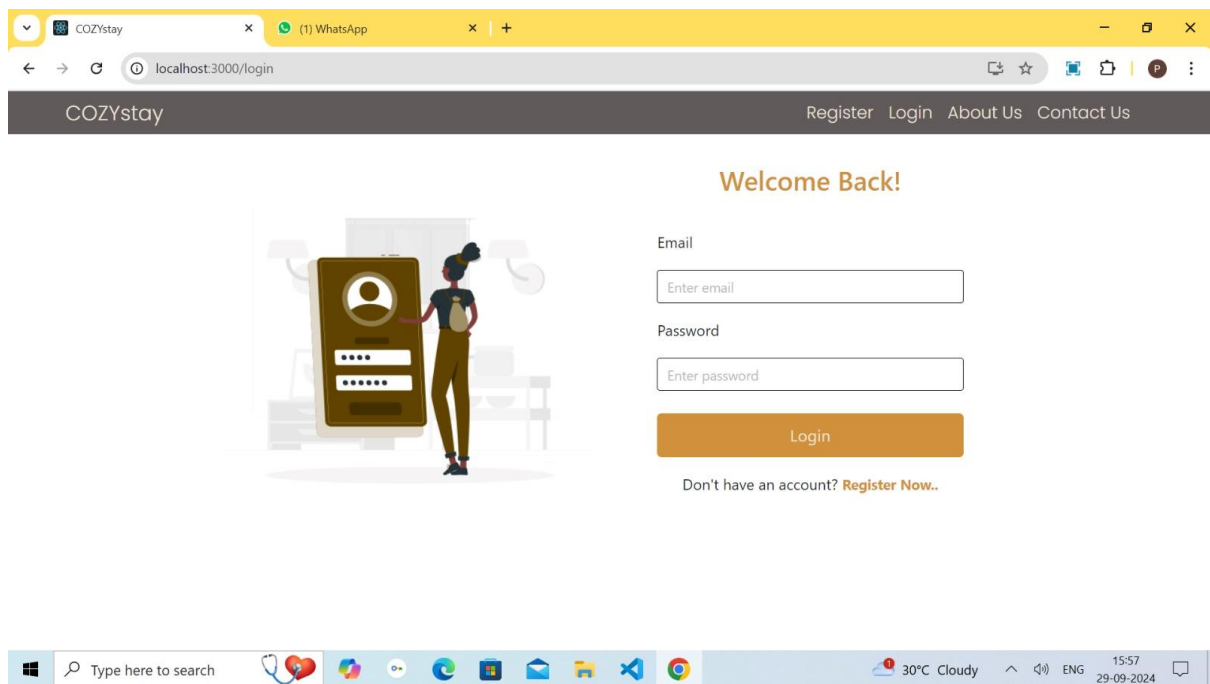
➤ landingScreen.jsx



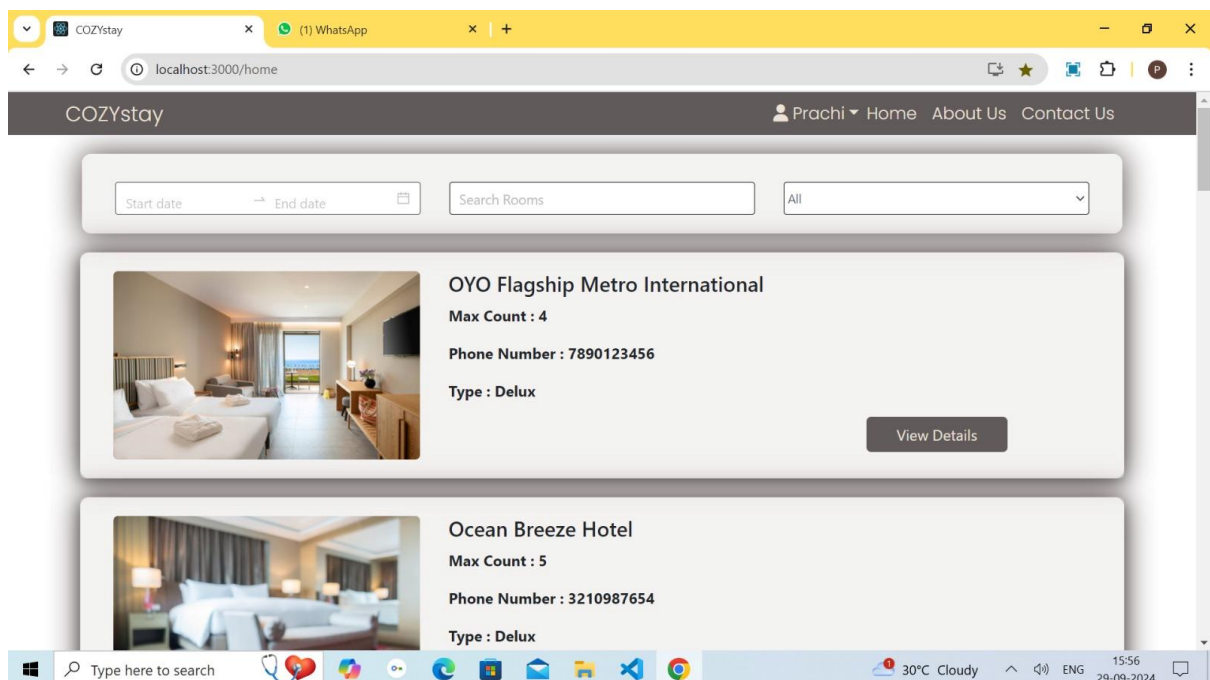
➤ registerScreen.jsx



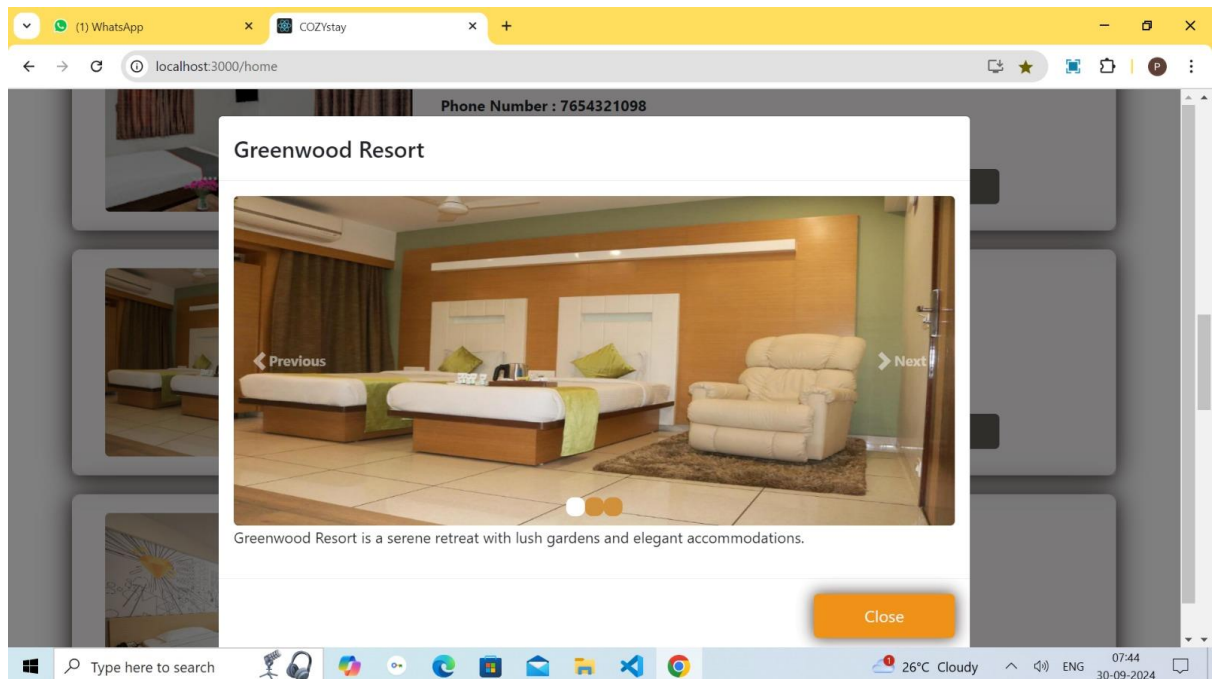
➤ loginScreen.jsx



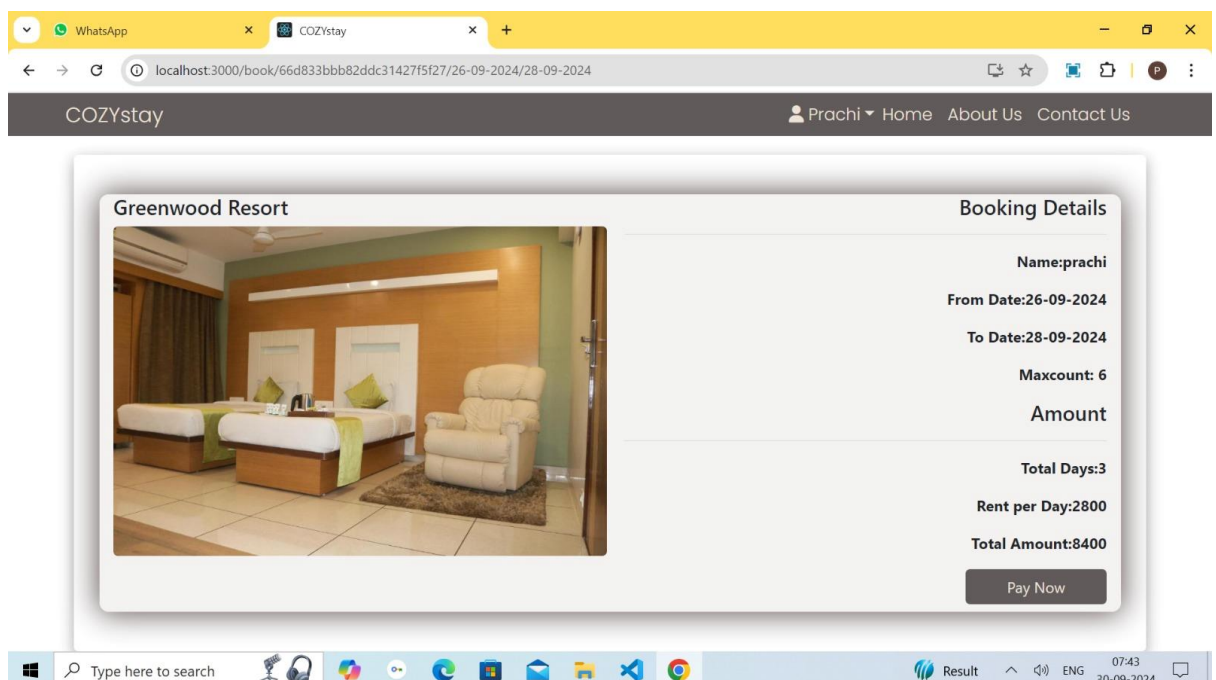
➤ homeScreen.jsx



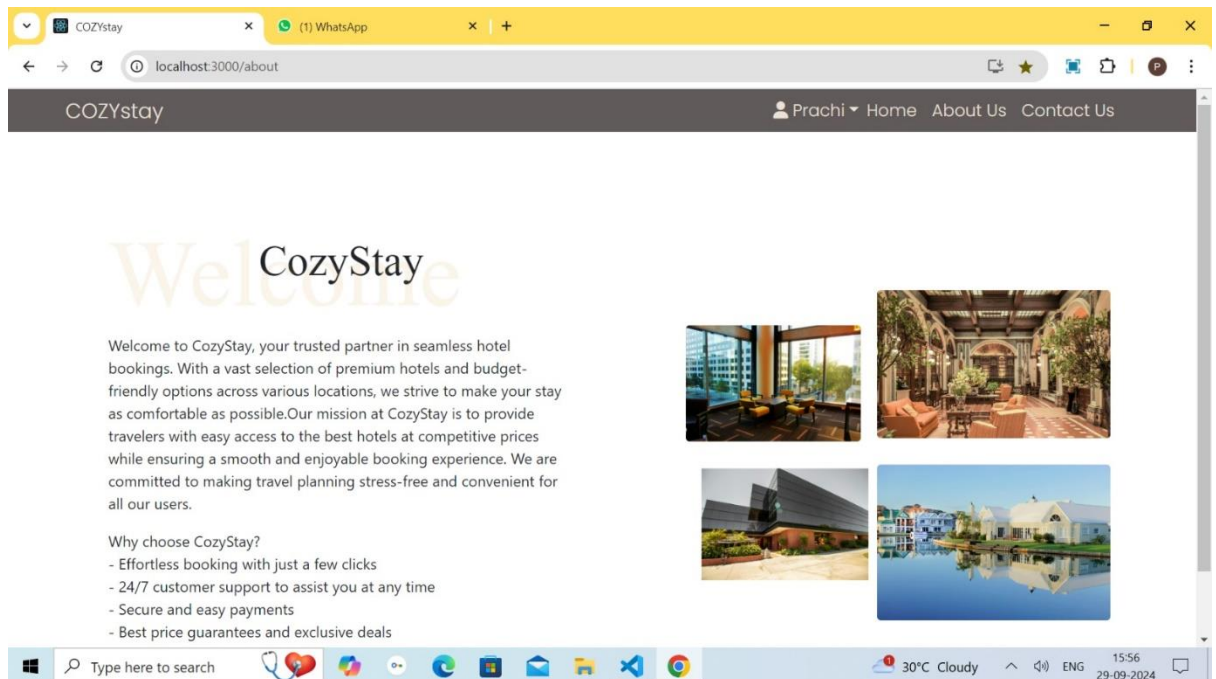
➤ room.jsx



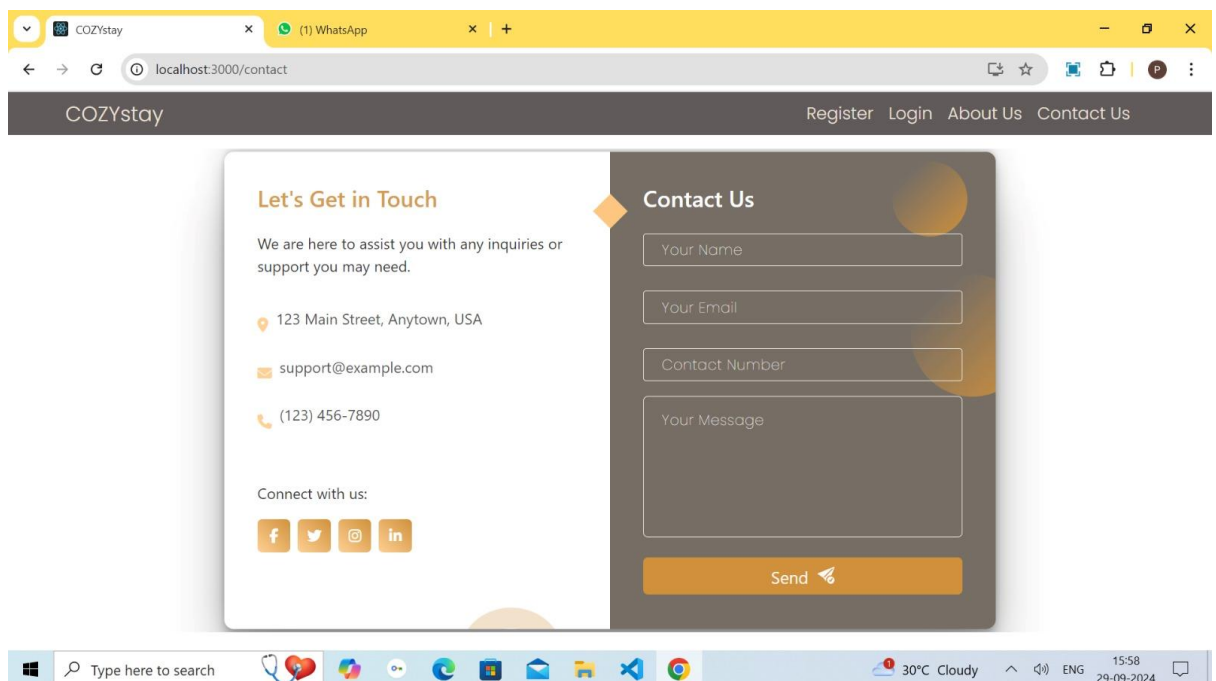
➤ bookingScreen.jsx



➤ aboutUs.jsx



➤ contactUs.jsx



7. LIMITATIONS AND FUTURE SCOPE OF ENHANCEMENTS

- The current hotel management systems face several limitations that impact their overall effectiveness. One significant challenge is integration; many systems struggle to connect seamlessly with existing legacy software and third-party applications.
- This can hinder the ability of hotels to fully utilize their technological investments.
- Additionally, the user interface of some systems can be outdated or unintuitive, making it difficult for staff to navigate and requiring extensive training.
- Moreover, the lack of customization options can prevent hotels from tailoring the system to their specific needs.
- Lastly, the quality of technical support and maintenance can vary widely, sometimes resulting in frustrating user experiences.

Future Scope of Enhancements:

- The future scope for enhancing hotel management systems is promising.
- Implement AI-driven features for personalized guest experiences, dynamic pricing, and predictive analytics.
- Develop mobile apps for staff and guests, allowing for on-the-go management and booking.
- Integrate features that help track and manage sustainability initiatives, such as energy consumption.
- Provide more customizable reporting tools and dashboards for different stakeholders.

8. REFERENCES

For React.js :

<https://react.dev/>

<https://legacy.reactjs.org/>

<https://www.npmjs.com/package/react-router-dom>

For Node.js :

<https://nodejs.org/en>

For MongoDB :

<https://www.mongodb.com/>

<https://www.mongodb.com/products/tools/compass>