Final Project Report

Hotel Booking System

CRN-31033: Group4

Presented By:

Sai Kiran (700724533),

Prathyusha (700728370),

Rupendra Varma Mudunuri(700725327),

Lathasri Rayarapu().

Project Overview

Description:

This Online Hotel Reservation System project aims at providing the user to reserve accommodation at hotels online. The system shall take the start and end dates from the user and check for availability of rooms. It shall check for the number of guests and reserve the rooms for the user. It can also modify the dates of reservation. This is a simple user interface which displays the information about the hotel, its contact address and the amenities at the hotel. It also provides the rates of rooms in that hotel. This tool shall enable the user to check for information regarding the hotel and reserve rooms. It enables the user to make payments online. And admin can also add the hotels and users as per requirement

Existing System:

The Reservation was not done online till now. The customers had to call up the hotel and make bookings depending on the availability. The information was given individually to all the customers who called for reservations. This was a difficult task.

Proposed Online Hotel Reservation System:

This Online Hotel Reservation System will help the user to make reservation for hotels online. The user can check the availability of the rooms. The tool provides an option to input start and end dates of the booking. It also takes input such as the number of guests. Depending on the availability of the rooms the user can book the room. The user also has an option to update the reservation details. The user can view the room rates. He can check for different amenities provided by the hotel. The user can also know more about the hotel and its address. Also, gallery provides pictures of different type of accommodations. This would ease the customer to book rooms. The tool has an option to make payments online. The Interface is simple which uses latest web technologies. It's well secure for payments

Online Hotel Reservation System Specifications Modules

Modules

Hotel: This module shall have all the details regarding the hotel such as name, number of rooms, type, address etc.

Admin: Here Admin can view or delete or update Rooms and Hotels.

Booking: This module shall have all the details regarding the start and end dates of reservation for different rooms.

Payment: This module holds the payment related information. It takes care of the payment for reservation online. It verifies and validates the payments done online.

Database Structure

Description

Hotel Booking System database includes six collection users, rooms, room_type, hotels, reservation, payment, transaction. Each user can book a hotel and select rooms based on their requirement. Each user has a username and password which he can use to login into the website. And he also has a phone number and email. There will be an admin user as well; We can identify who is the user by using the field User type. Admins have access to add or delete user and also add or delete a hotel from the website. And coming to the hotel collection, hotels will have the following mandatory fields, hotel_name, hotel_type, city, address, distance, title, description of the hotel. As each hotel will be having different prices.

For each hotel, we must keep track of which user booked the particular hotel and particular room, payment information, and the date and time of the hotel booked. So, all the details related to booking is included in the Reservation Collection. Even check in, check out, and guest list are included in the Reservation collection.

And whenever a particular room is booked in the hotel, it should not be visible to the other customer who is willing to book that hotel, so we will be marking the hotel booked as unavailable in Availability field in Rooms collections. There are also few fields in Rooms collection like title, price, capacity, facilities and availability, these details are very much important for the user to decide what room to book based on his requirement.

Payment Information is stored in Payment collection with unique transaction id. So here we are implementing whole project using MERN (MangoDB, Express, React, Nodejs) Stack.

Data Dictionary

Collection 1- User Details: Contains all the user's information.

- User user_id
- User firstname
- User lastname
- User email
- User Phone number
- User Location
- User type

Collection 2- Hotel Details: Contains all the hotel's details.

- Hotel id
- Hotel Name
- Hotel location
- Hotel owner

Collection 3- Room Details: Contains the room details for the selected Hotel.

- Room room_id
- Room hotel_id
- Room capacity
- Room price
- Room facilities
- Room Availability

Collection 4- Reservation Details: Contains the reservation details of the user and the booked hotel.

- reservation _id
- hotel_id
- User_id
- Room_type_id
- Guest_list
- Check_in
- Checkout
- Balanceamount

Collectoin 5- room_type: Contains the type of room type, price and number of guest details.

- Room_type_id
- Price
- Capacity

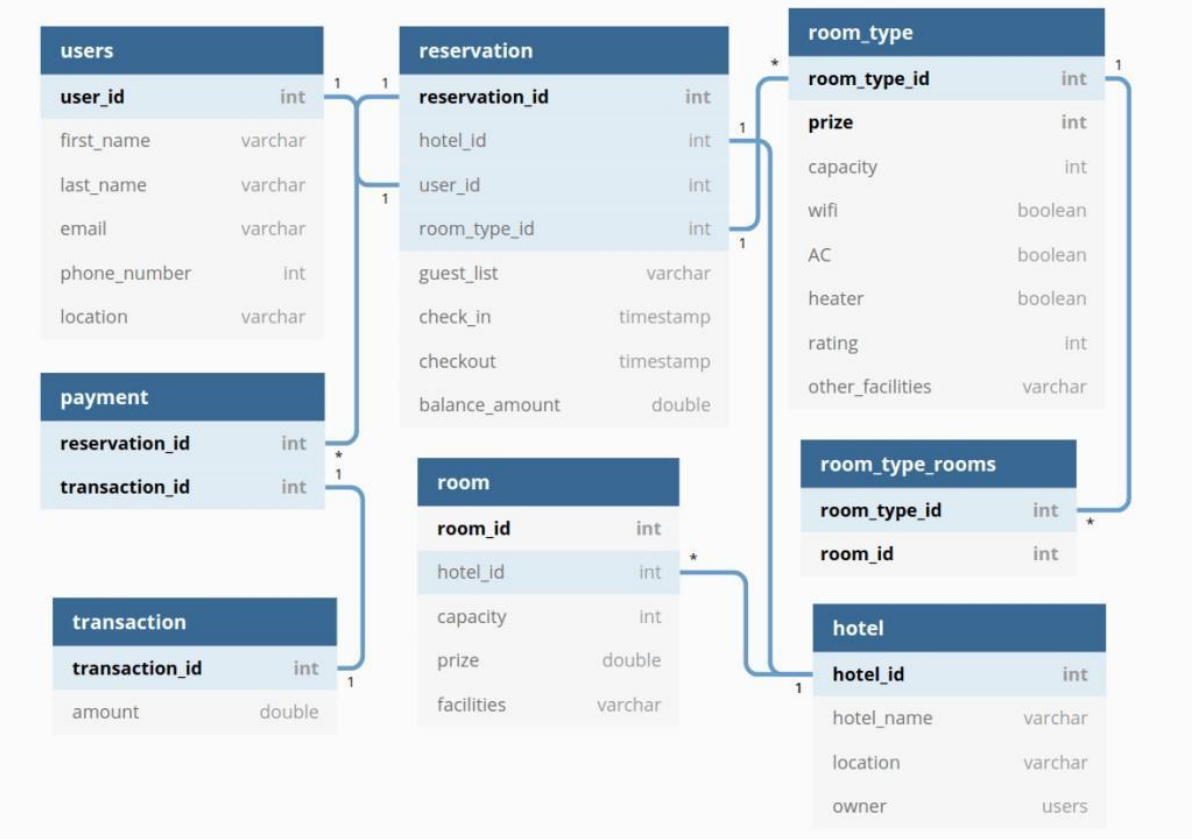
Collection 6- Payment: Contains the transaction details of the reservation.

- Reservation_id
- Transcation_id

Collection 7- Transaction: Contains the amount details of the transaction.

- Transaction_id
- Amount

Database diagram



Sample data

User Details:

_id:629e01fd6a52a0f0dee31
User_id: 101
first_name:"Steven"
last_name:"Chris"
email:"Chris@gmail.com"
Phonenumber:+19139139139
Location:"8000 W 148th St"
featured:false
__v:0

_id:629a01ff4a52a0f0dee311d3
User_id: 102
first_name:"Graham"
last_name:"William"
email:"william@gmail.com"
Phonenumber:+19139234559
Location:"8002 W 132nd St"
featured:false
__v:0

Hotel Details:

_id:629d03fd6a52a0f0dee311d3
Hotel_id: 11
Hotel_name:"Steven"
Owner: Ramu
Location:"8000 W 148th St"
featured:false
__v:0

_id:629d03fd6a52a0f0dee311d3
Hotel_id: 14
Hotel_name:"Raaga"
Owner: Chris
Location:"8000 W 148th St"
featured:false
__v:0

Room Details:

_id:629a01ff4a52a0f0dee311d3
Room_id: 101
Hotel_id:11
Capacity : 4
facilities :"Luxury room with 3 beds"
featured:false
__v:0

_id:629a01ff4a52a0f0dee311d3
Room_id: 102
Hotel_id:14
Capacity : 4
facilities :"Luxury room with 3 beds"
featured:false
__v:0

Reservation Details:

_id:629a01ff4a52a0f0dee311d3
Reservation_id:bb1234 \
Hotel_id:11
User_id:101
Room_type_id:bb11
guest_list:[]
check_in:12/1/2022
check_out:12/5/2022
__v:0

Room_type Details:

_id:629a01ff4a52a0f0dee456d3
room_type_id_id:bb123
room_id:101

Payment Details:

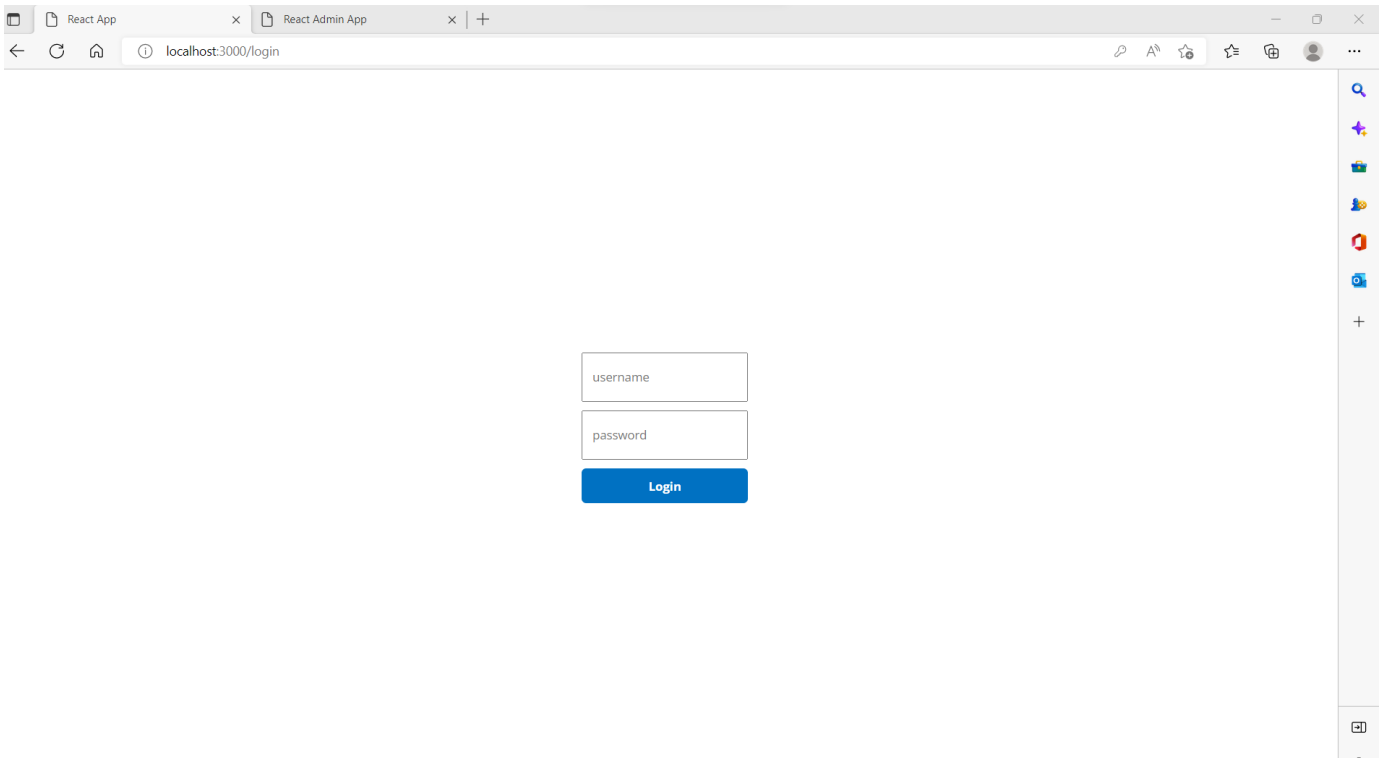
_id:629a01ff4a52a0f0dee232d3
Reservation_id :bb123
Transaction_id :101

Transaction Details:

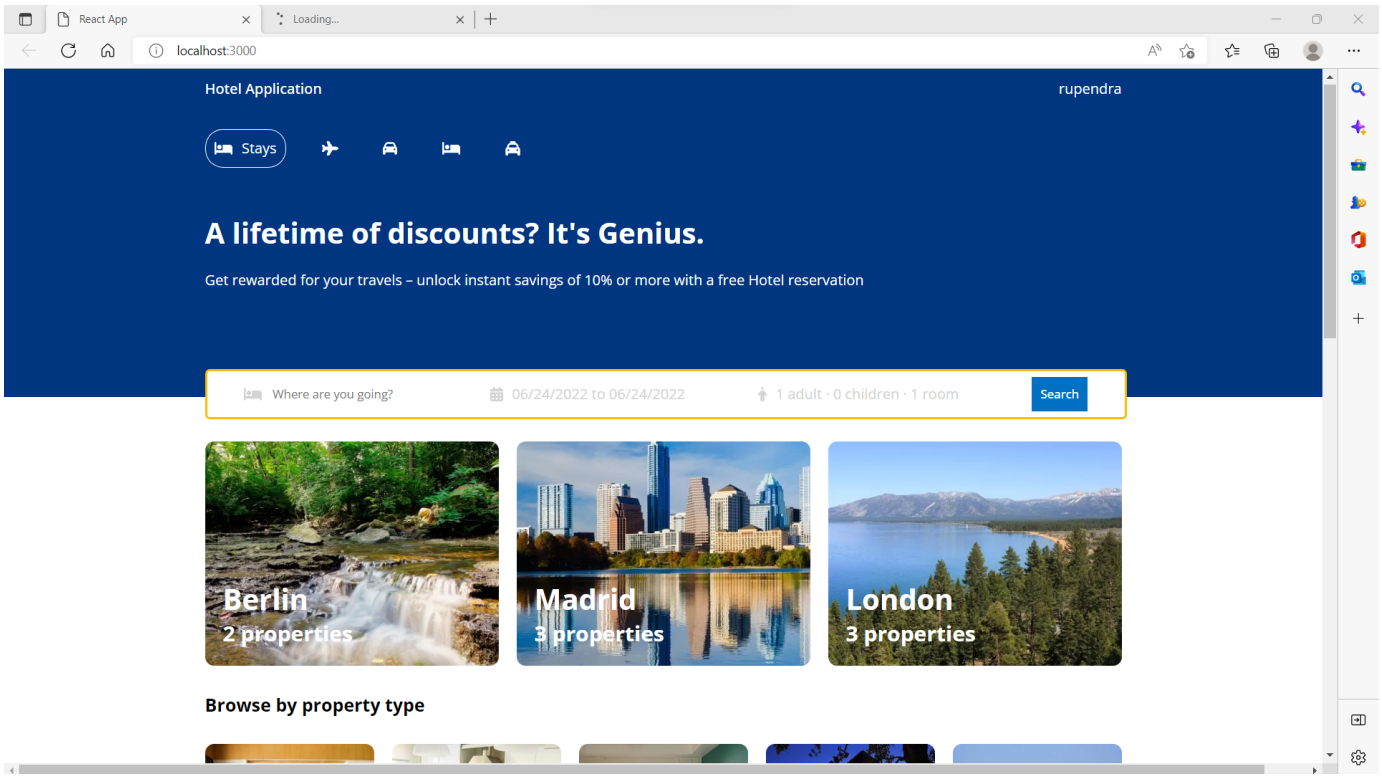
_id:629a01ff4a52a0f0dee232d3
Transaction_id :101
Amount :200$

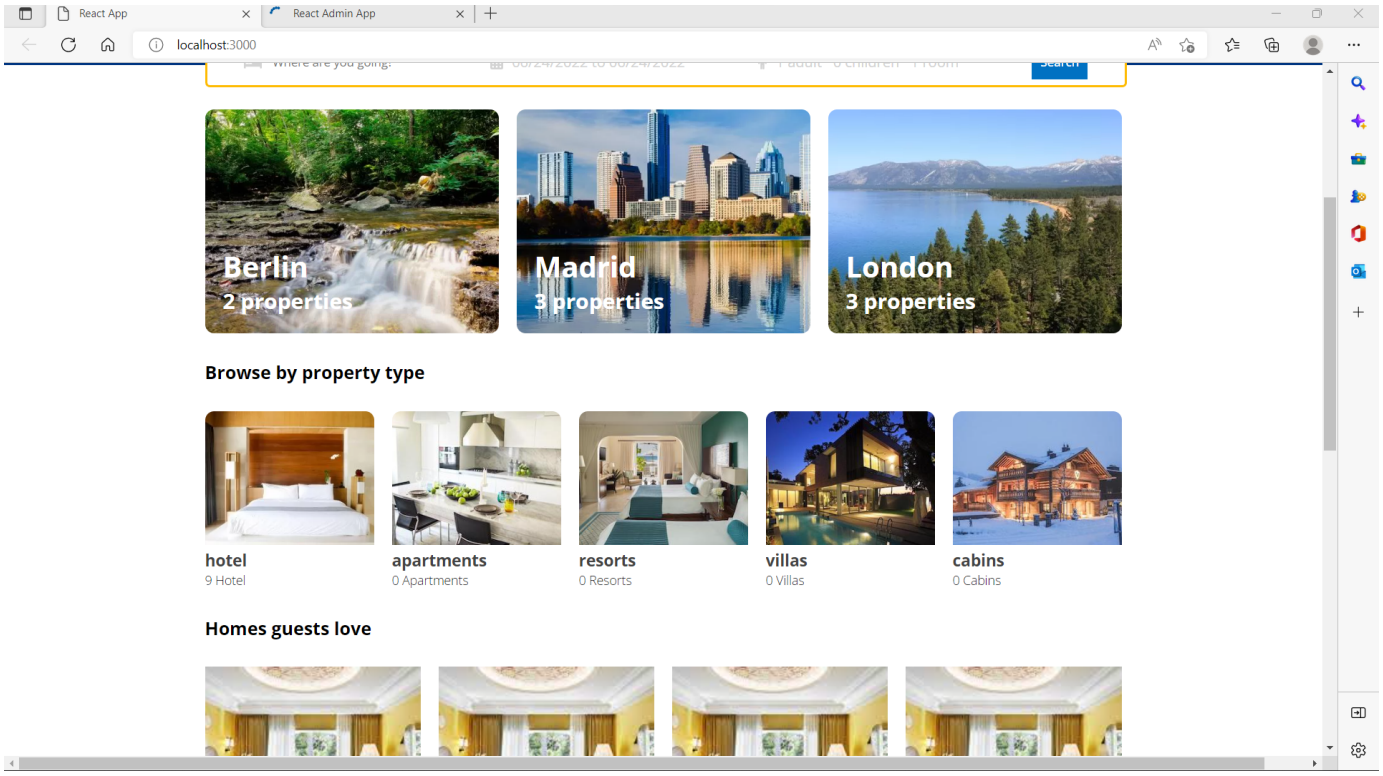
User Interfaces and Forms:

User & Admin Login Page:

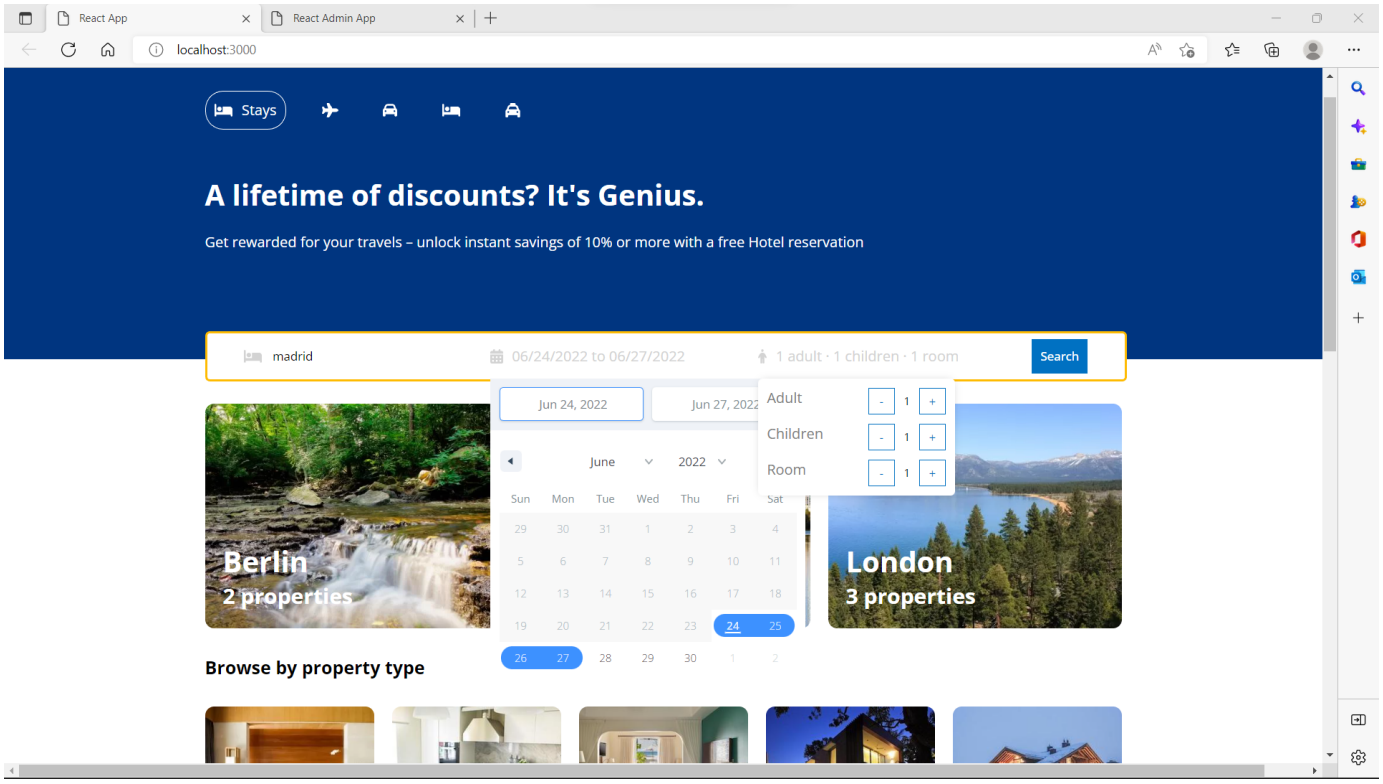


Home Page:

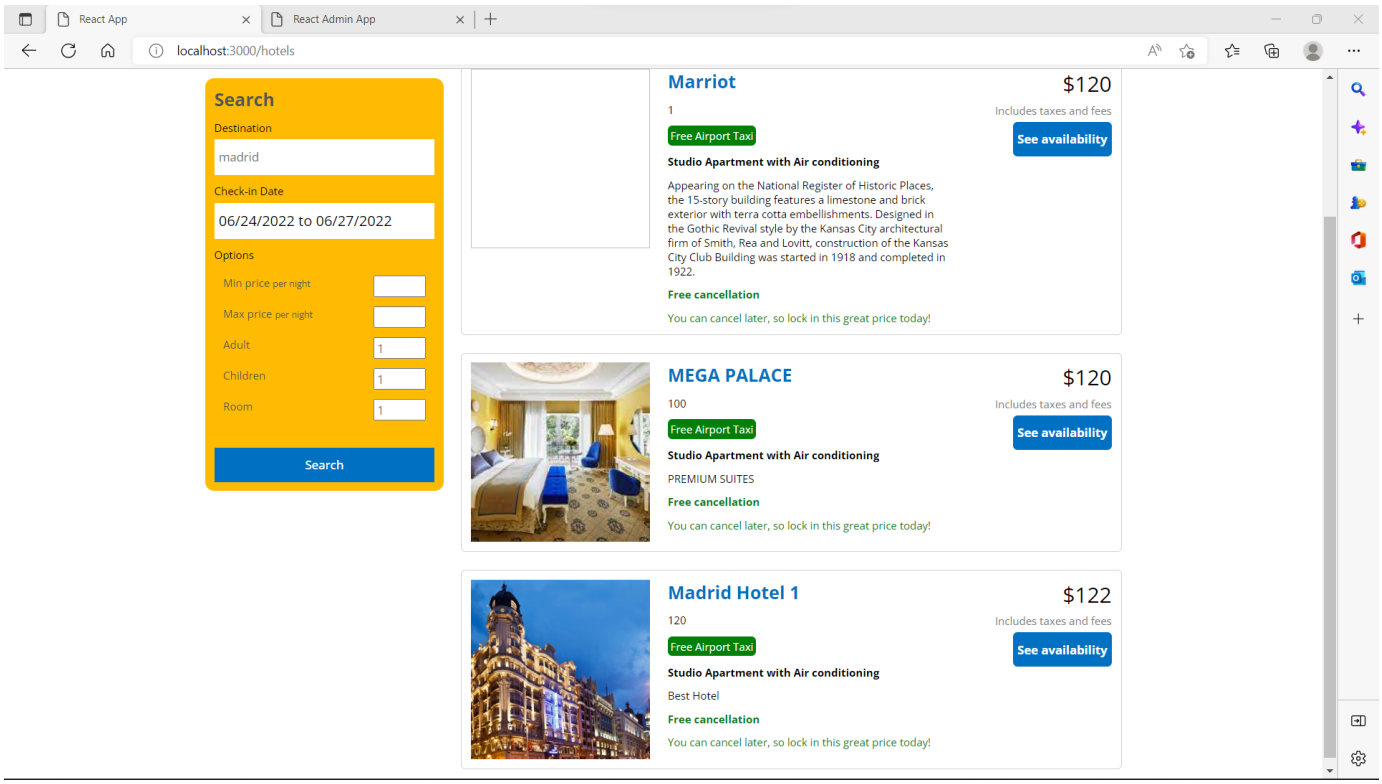




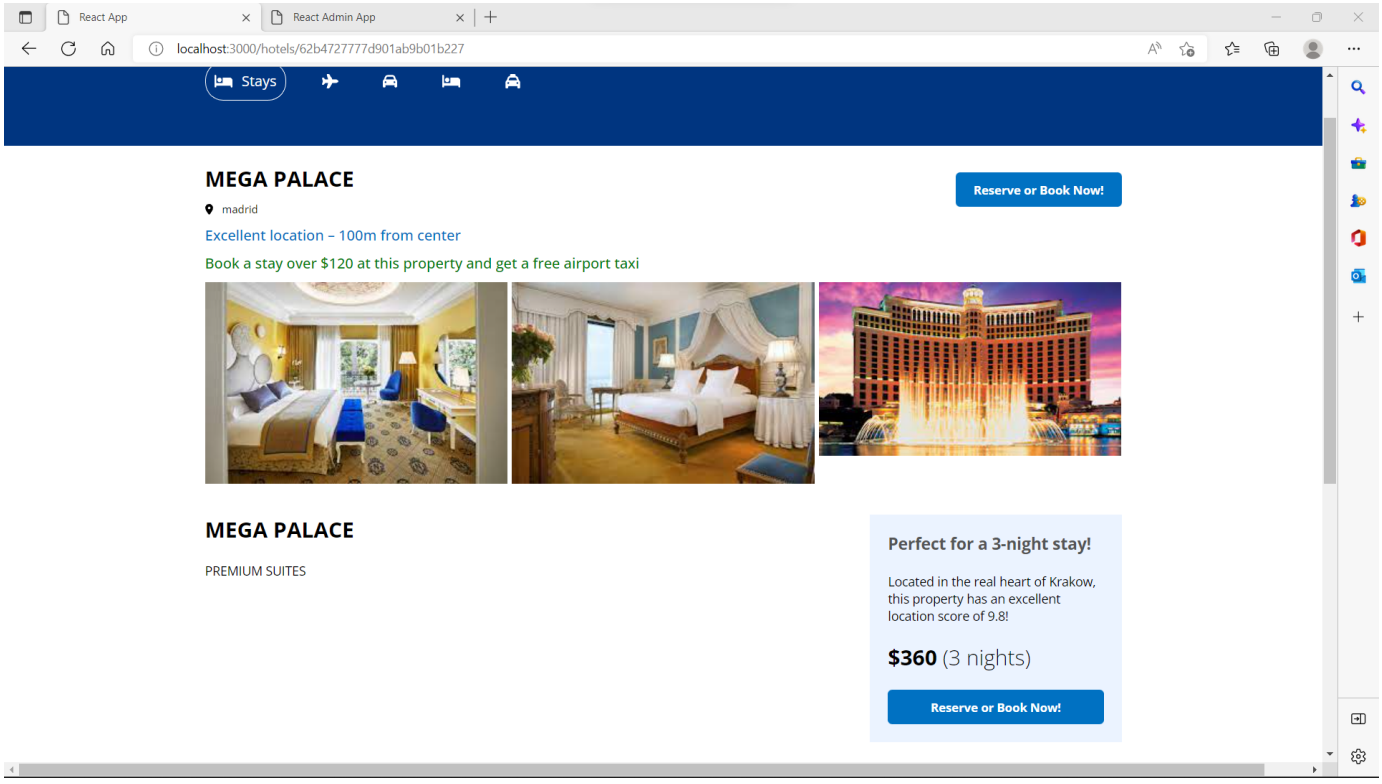
Hotel Search Item: You can search your hotel based on your requirements.



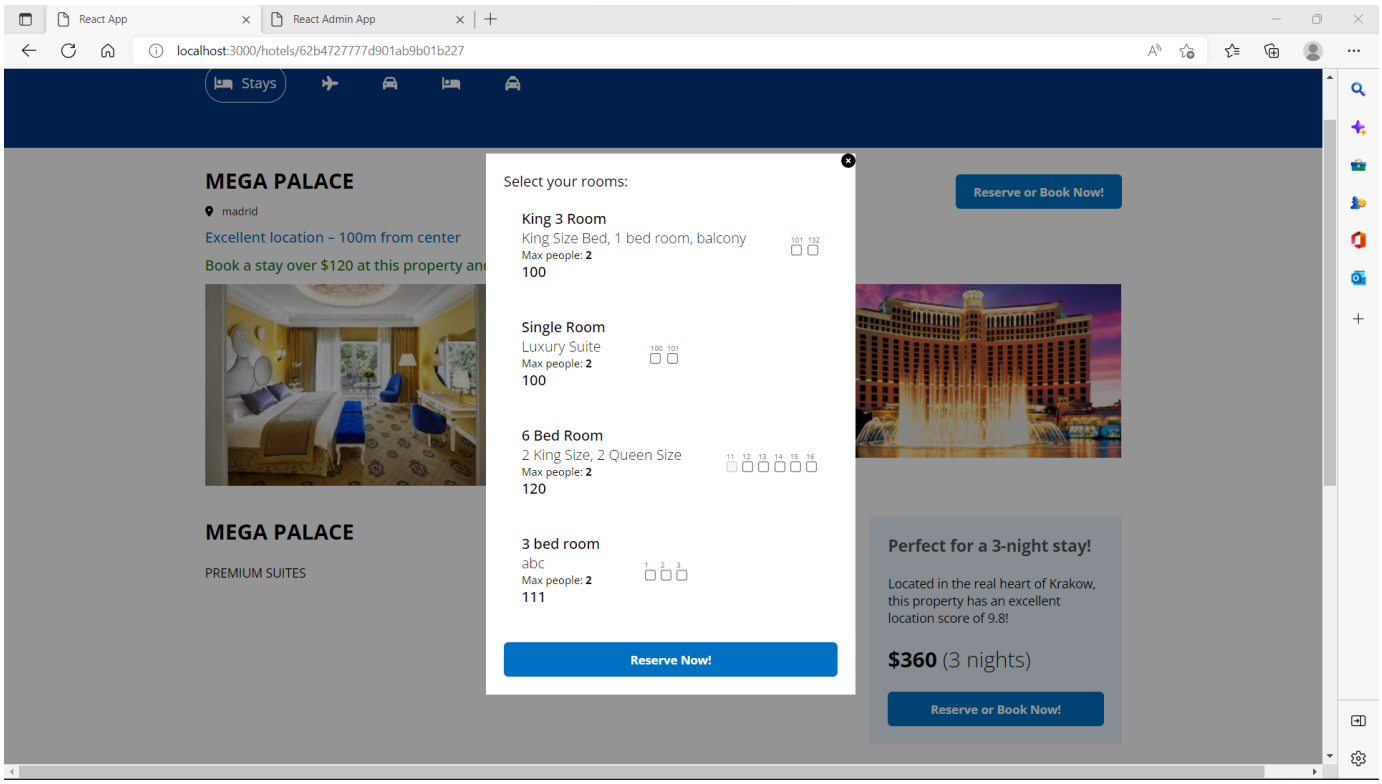
Hotel Search Results:



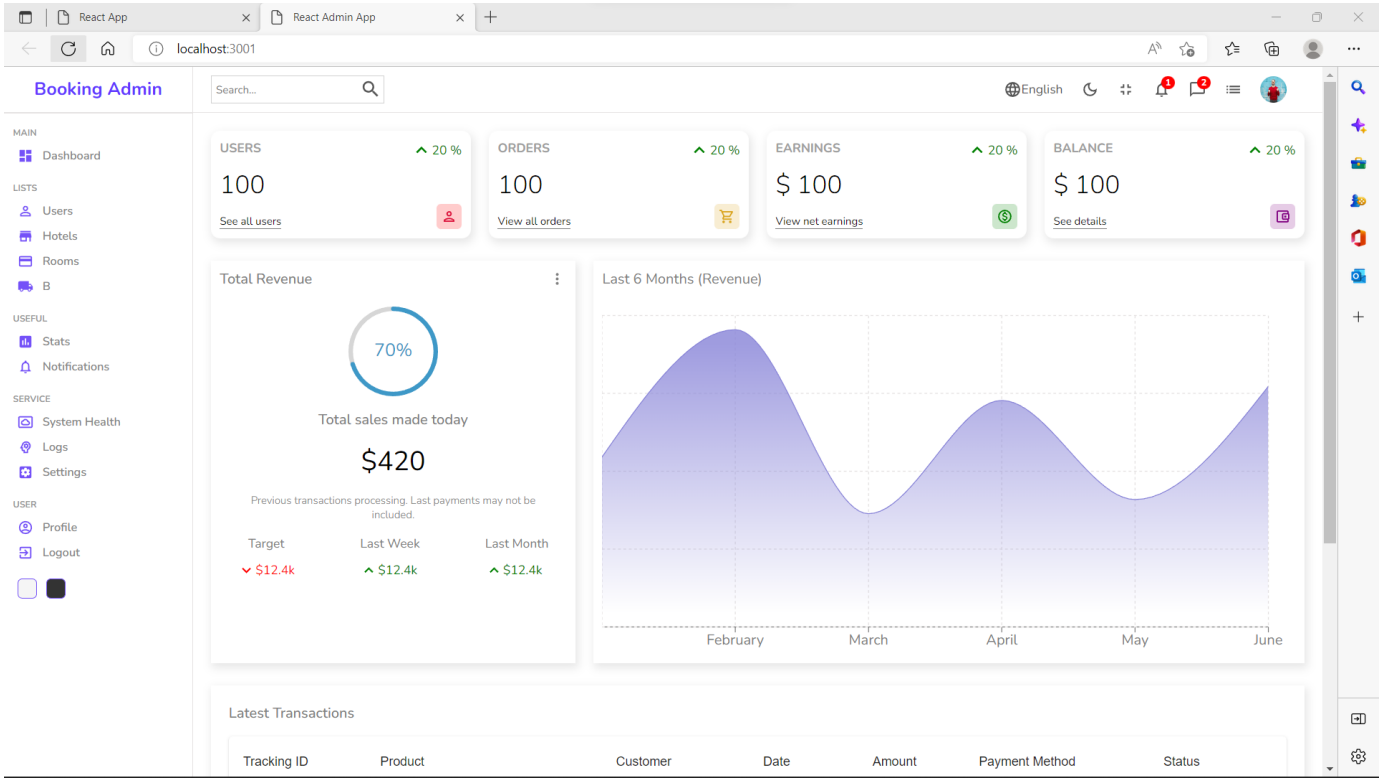
Hotel Reservation Component: You can reserve by clicking on reserve



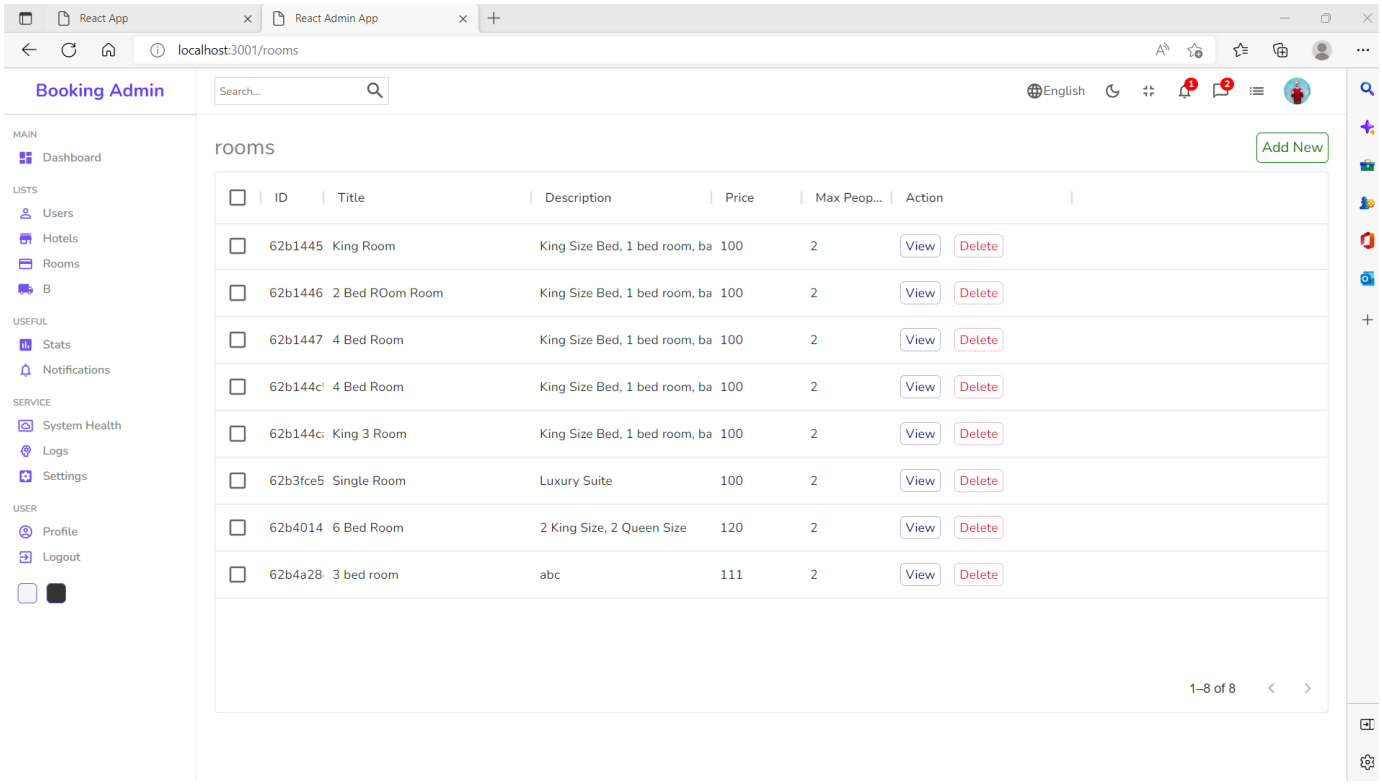
Hotel Room Selection: While reserving hotel you will prompted with select rooms and reserve options



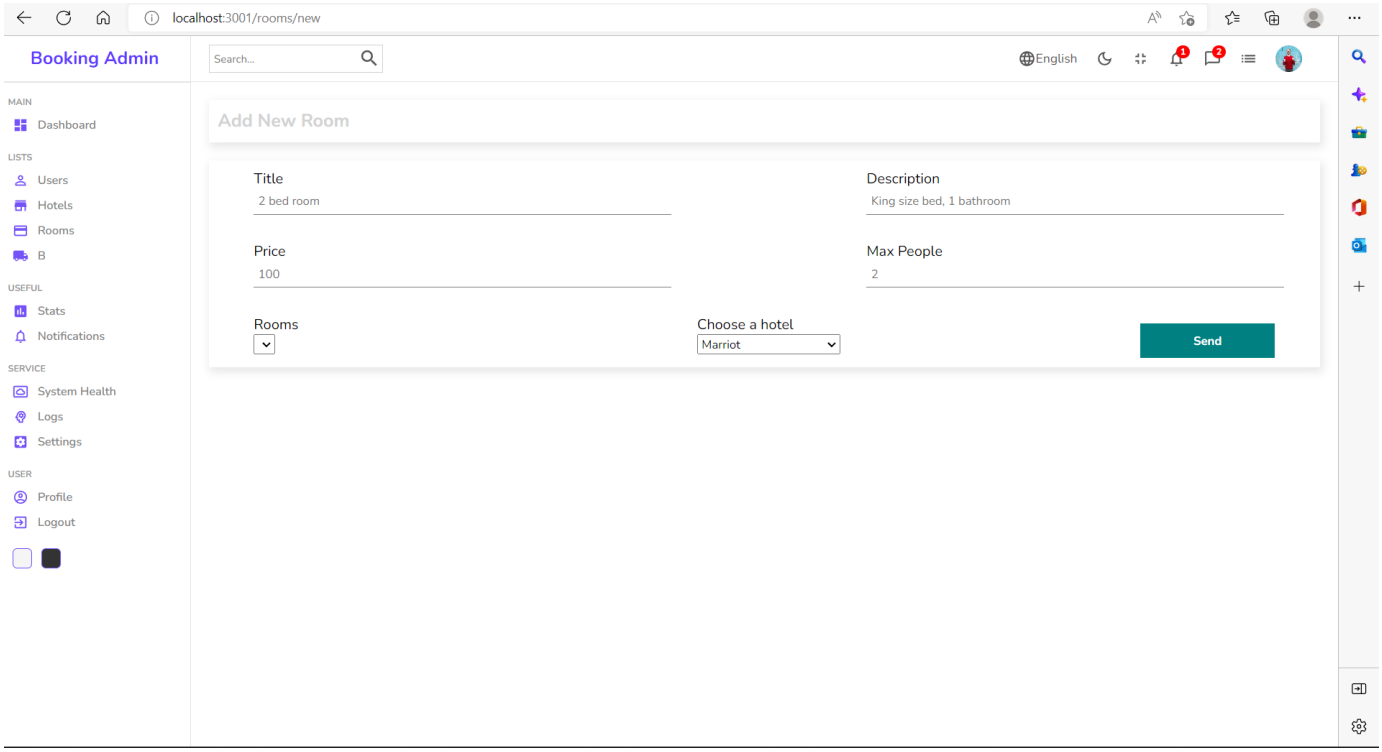
Admin Home Panel:



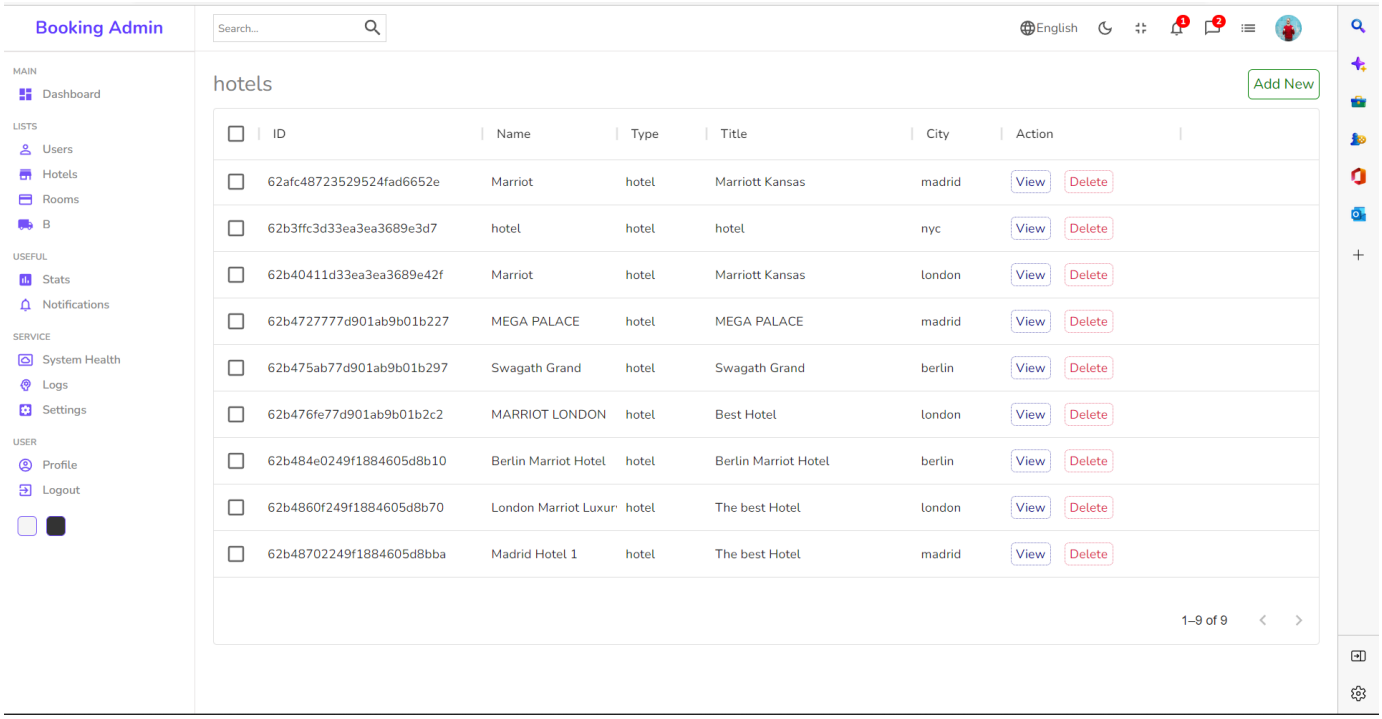
Admin Rooms component:

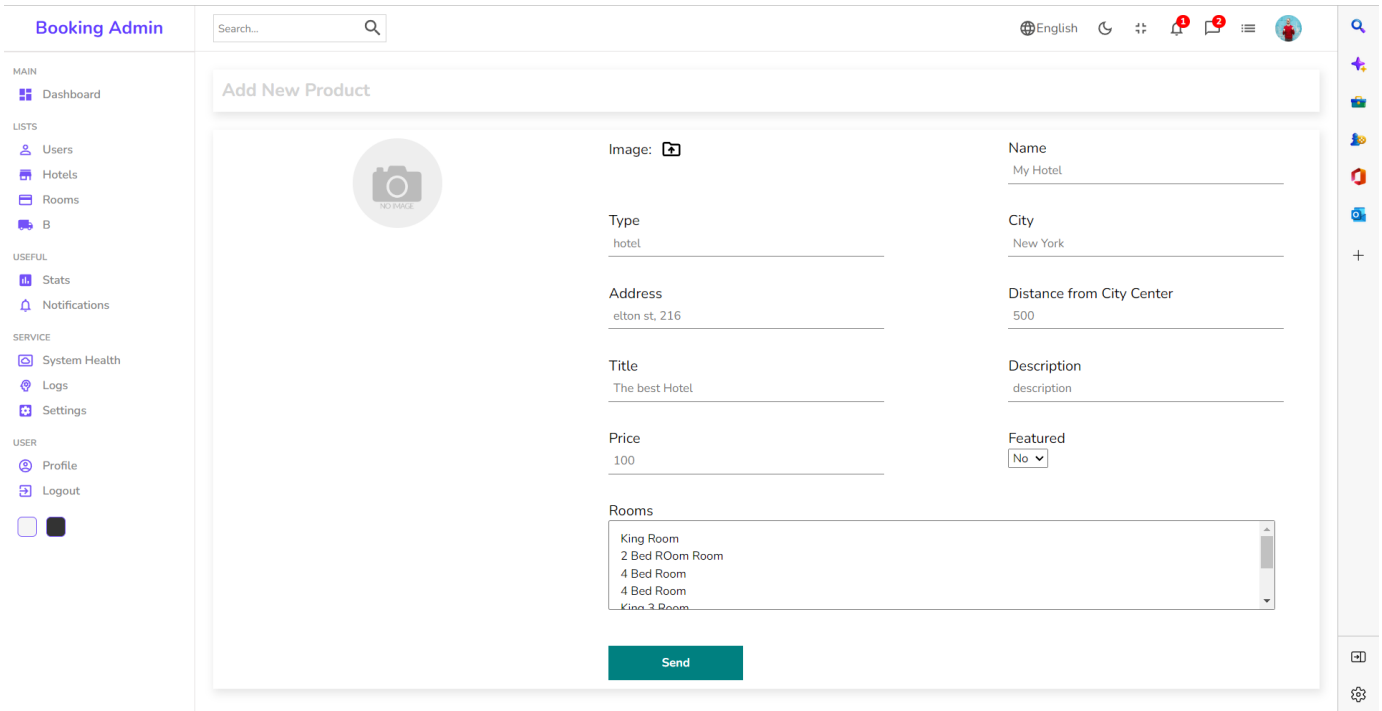


Admin Room Add or Delete:

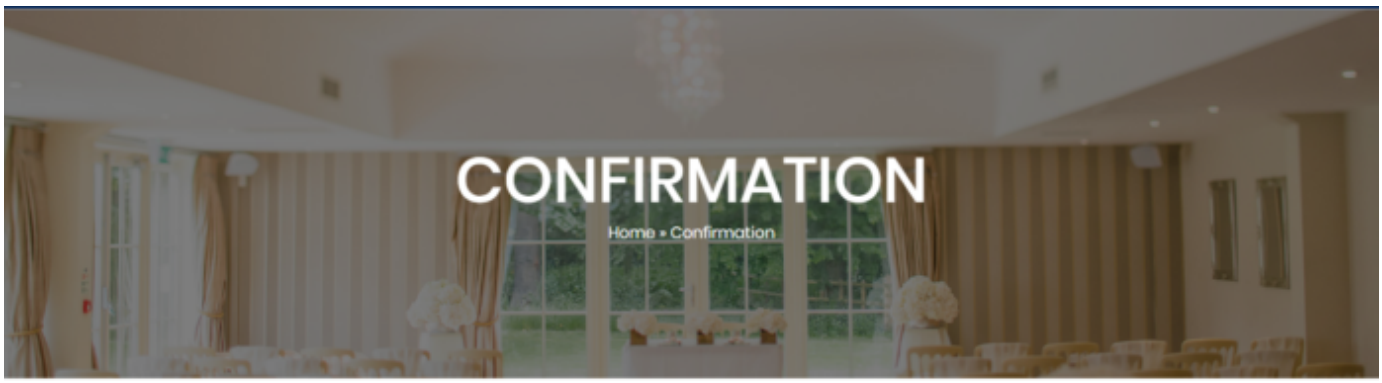


Admin Hotel Add or Delete:





Booking Confiramtion Page:



1. Choose Room 2. Make a Reservation 3. Confirmation

YOUR RESERVATION

Order #190424-195533531

DeLuxe room

Check In:	April 27, 2019	
Check Out:	April 28, 2019	
Guests:	Adults	2
Services:	Wedding venue	$10

Total amount: $120

Paid: $0

Email:	sergey.traveler@gmail.com
First name:	John
Last name:	Smith
Contact Phone:	1234567890

RESERVATION COMPLETE

Your order is completed and received, and a confirmation email was sent to you. You will pay the full amount later. Thank you!

Source Codes:

Backend Code: api/index.js

```
import express from "express"
import dotenv from "dotenv"
import mongoose from "mongoose"
import authRoute from "./routes/auth.js";
import usersRoute from "./routes/users.js";
import hotelsRoute from "./routes/hotels.js";
import roomsRoute from "./routes/rooms.js";
import cookieParser from "cookie-parser";
import cors from "cors"



const app = express()
dotenv.config()

const connect = async () => {
    try {
      await mongoose.connect(process.env.MONGO);
      console.log("Connected to mongoDB.");
    } catch (error) {
      throw error;
    }
  };

  mongoose.connection.on("disconnected", () => {
    console.log("mongoDB disconnected!");
  });


//middlewares

app.use(cookieParser())
app.use(cors())
app.use(express.json())
app.use("/api/auth", authRoute);
app.use("/api/users", usersRoute);
app.use("/api/hotels", hotelsRoute);
app.use("/api/rooms", roomsRoute);
```

```
app.use((err, req, res, next) => {
  const errorStatus = err.status || 500;
  const errorMessage = err.message || "Something went wrong!";
  return res.status(errorStatus).json({
    success: false,
    status: errorStatus,
    message: errorMessage,
    stack: err.stack,
  });
});


app.listen(8800, () => {
    connect()
    console.log("Connected to backend.");
  });
```

api/controllers:

```
import User from "../models/User.js";
import bcrypt from "bcryptjs";
import { createError } from "../utils/error.js";
import jwt from "jsonwebtoken";

export const register = async (req, res, next) => {
  try {
    const salt = bcrypt.genSaltSync(10);
    const hash = bcrypt.hashSync(req.body.password, salt);

    const newUser = new User({
      ...req.body,
      password: hash,
    });

    await newUser.save();
    res.status(200).send("User has been created.");
  } catch (err) {
    next(err);
  }
```

```
};
export const login = async (req, res, next) => {
  try {
    const user = await User.findOne({ username: req.body.username });
    if (!user) return next(createError(404, "User not found!"));

    const isPasswordCorrect = await bcrypt.compare(
      req.body.password,
      user.password
    );
    if (!isPasswordCorrect)
      return next(createError(400, "Wrong password or username!"));

    const token = jwt.sign(
      { id: user._id, isAdmin: user.isAdmin },
      process.env.JWT
    );

    const { password, isAdmin, ...otherDetails } = user._doc;
    res
      .cookie("access_token", token, {
        httpOnly: true,
      })
      .status(200)
      .json({ details: { ...otherDetails }, isAdmin });
  } catch (err) {
    next(err);
  }
};
```

api/controllers/hotel.js

```
import Hotel from "../models/Hotel.js";
import Room from "../models/Room.js";

export const createHotel = async (req, res, next) => {
  const newHotel = new Hotel(req.body);

  try {
    const savedHotel = await newHotel.save();
    res.status(200).json(savedHotel);
  } catch (err) {
    next(err);
```

```
  }
};
export const updateHotel = async (req, res, next) => {
  try {
    const updatedHotel = await Hotel.findByIdAndUpdate(
      req.params.id,
      { $set: req.body },
      { new: true }
    );
    res.status(200).json(updatedHotel);
  } catch (err) {
    next(err);
  }
};
export const deleteHotel = async (req, res, next) => {
  try {
    await Hotel.findByIdAndDelete(req.params.id);
    res.status(200).json("Hotel has been deleted.");
  } catch (err) {
    next(err);
  }
};
export const getHotel = async (req, res, next) => {
  try {
    const hotel = await Hotel.findById(req.params.id);
    res.status(200).json(hotel);
  } catch (err) {
    next(err);
  }
};
export const getHotels = async (req, res, next) => {
  const { min, max, ...others } = req.query;
  try {
    const hotels = await Hotel.find({
      ...others,
      cheapestPrice: { $gt: min | 1, $lt: max || 999 },
    }).limit(req.query.limit);
    res.status(200).json(hotels);
  } catch (err) {
    next(err);
  }
};
export const countByCity = async (req, res, next) => {
```

```
  const cities = req.query.cities.split(",");
  try {
    const list = await Promise.all(
      cities.map((city) => {
        return Hotel.countDocuments({ city: city });
      })
    );
    res.status(200).json(list);
  } catch (err) {
    next(err);
  }
};
export const countByType = async (req, res, next) => {
  try {
    const hotelCount = await Hotel.countDocuments({ type: "hotel" });
    const apartmentCount = await Hotel.countDocuments({ type: "apartment" });
    const resortCount = await Hotel.countDocuments({ type: "resort" });
    const villaCount = await Hotel.countDocuments({ type: "villa" });
    const cabinCount = await Hotel.countDocuments({ type: "cabin" });

    res.status(200).json([
      { type: "hotel", count: hotelCount },
      { type: "apartments", count: apartmentCount },
      { type: "resorts", count: resortCount },
      { type: "villas", count: villaCount },
      { type: "cabins", count: cabinCount },
    ]);
  } catch (err) {
    next(err);

  }
};

export const getHotelRooms = async (req, res, next) => {
  try {
    const hotel = await Hotel.findById(req.params.id);
    const list = await Promise.all(
      hotel.rooms.map((room) => {
        return Room.findById(room);
      })
    );
    res.status(200).json(list)
  } catch (err) {
```

```
    next(err);
  }
};
```

api/controllers/room.js

```
import Room from "../models/Room.js";
import Hotel from "../models/Hotel.js";
import { createError } from "../utils/error.js";

export const createRoom = async (req, res, next) => {
  const hotelId = req.params.hotelid;
  const newRoom = new Room(req.body);

  try {
    const savedRoom = await newRoom.save();
    try {
      await Hotel.findByIdAndUpdate(hotelId, {
        $push: { rooms: savedRoom._id },
      });
    } catch (err) {
      next(err);
    }
    res.status(200).json(savedRoom);
  } catch (err) {
    next(err);
  }
};

export const updateRoom = async (req, res, next) => {
  try {
    const updatedRoom = await Room.findByIdAndUpdate(
      req.params.id,
      { $set: req.body },
      { new: true }
    );
    res.status(200).json(updatedRoom);
  } catch (err) {
    next(err);
  }
};
export const updateRoomAvailability = async (req, res, next) => {
  try {
```

```
    await Room.updateOne(
      { "roomNumbers._id": req.params.id },
      {
        $push: {
          "roomNumbers.$.unavailableDates": req.body.dates
        },
      }
    );
    res.status(200).json("Room status has been updated.");
  } catch (err) {
    next(err);
  }
};
export const deleteRoom = async (req, res, next) => {
  const hotelId = req.params.hotelid;
  try {
    await Room.findByIdAndDelete(req.params.id);
    try {
      await Hotel.findByIdAndUpdate(hotelId, {
        $pull: { rooms: req.params.id },
      });
    } catch (err) {
      next(err);
    }
    res.status(200).json("Room has been deleted.");
  } catch (err) {
    next(err);
  }
};
export const getRoom = async (req, res, next) => {
  try {
    const room = await Room.findById(req.params.id);
    res.status(200).json(room);
  } catch (err) {
    next(err);
  }
};
export const getRooms = async (req, res, next) => {
  try {
    const rooms = await Room.find();
    res.status(200).json(rooms);
  } catch (err) {
    next(err);
```

```
  }
};
```

api/controllers/user.js

```
import User from "../models/User.js";


export const updateUser = async (req, res, next) => {

  try{
    const updatedUser = await User.findByIdAndUpdate(req.params.id, { $set:
req.body}, {new: true})
    res.status(200).json(updatedUser)
}catch(err){
    next(err);
  }
};
export const deleteUser = async (req, res, next) => {

  try{
    await User.findByIdAndDelete(
        req.params.id
    );
    res.status(200).json("User has been deleted")
}catch(err){
    next(err);
  }
};
export const getUser = async (req, res, next) => {

  try{
    const user = await User.findById(req.params.id);
    res.status(200).json(user)
}catch(err){
    next(err);
  }
};
export const getUsers = async (req, res, next) => {

  try{
    const users = await User.find();
    res.status(200).json(users)
```

```
}catch(err){
    next(err);
  }
};
```

api/routes/auth.js

```
import express from "express";
import { login, register } from "../controllers/auth.js";

const router = express.Router();

router.post("/register", register)



router.post("/login",login)

export default router;
```

api/routes/hotels.js

```
import express from "express";
import {
  countByCity,
  countByType,
  createHotel,
  deleteHotel,
  getHotel,
  getHotelRooms,
  getHotels,
  updateHotel,
} from "../controllers/hotel.js";
import Hotel from "../models/Hotel.js";
import {verifyAdmin} from "../utils/verifyToken.js"
const router = express.Router();

//CREATE
router.post("/", verifyAdmin, createHotel);
```

```
//UPDATE
router.put("/:id", verifyAdmin, updateHotel);
//DELETE
router.delete("/:id", verifyAdmin, deleteHotel);
//GET

router.get("/find/:id", getHotel);
//GET ALL

router.get("/", getHotels);
router.get("/countByCity", countByCity);
router.get("/countByType", countByType);
router.get("/room/:id", getHotelRooms);

export default router;
```

client/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { AuthContextProvider } from './context/AuthContext';
import { SearchContextProvider } from './context/SearchContext';

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



  </React.StrictMode>
);
```

client/app.js

```
import {
  BrowserRouter,
  Routes,
  Route,
} from "react-router-dom";
import Home from "./pages/home/Home";
import Hotel from "./pages/hotel/Hotel";
import List from "./pages/list/List";
import Login from "./pages/login/Login";


function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home/>}/>
        <Route path="/hotels" element={<List/>}/>
        <Route path="/hotels/:id" element={<Hotel/>}/>
        <Route path="/login" element={<Login/>}/>

      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

client/home.js

```
import Featured from "../../components/featured/Featured";
import FeaturedProperties from
"../../components/featuredProperties/FeaturedProperties";
import Footer from "../../components/footer/Footer";
import Header from "../../components/header/Header";
import MailList from "../../components/mailList/MailList";
import Navbar from "../../components/navbar/Navbar";
import PropertyList from "../../components/propertyList/PropertyList";
import "./home.css";

const Home = () => {
  return (
```

```
    <div>
      <Navbar />
      <Header/>
      <div className="homeContainer">
        <Featured/>
        <h1 className="homeTitle">Browse by property type</h1>
        <PropertyList/>
        <h1 className="homeTitle">Homes guests love</h1>
        <FeaturedProperties/>
        <MailList/>
        <Footer/>
      </div>
    </div>
  );
};

export default Home;
```

client/hotel.js

```
import "./hotel.css";
import Navbar from "../../components/navbar/Navbar";
import Header from "../../components/header/Header";
import MailList from "../../components/mailList/MailList";
import Footer from "../../components/footer/Footer";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import {
  faCircleArrowLeft,
  faCircleArrowRight,
  faCircleXmark,
  faLocationDot,
} from "@fortawesome/free-solid-svg-icons";
import { useContext, useState } from "react";
import useFetch from "../../hooks/useFetch";
import { useLocation, useNavigate } from "react-router-dom";
import { AuthContext } from "../../context/AuthContext";
import { SearchContext } from "../../context/SearchContext";
import Reserve from "../../components/reserve/Reserve";


const Hotel = () => {
  const location = useLocation();
  const id = location.pathname.split("/")[2];
```

```
const [slideNumber, setSlideNumber] = useState(0);
const [open, setOpen] = useState(false);
const [openModal, setOpenModal] = useState(false);

const { data, loading, error } = useFetch(`/hotels/find/${id}`);
const { user } = useContext(AuthContext);

const navigate = useNavigate();

const { dates, options } = useContext(SearchContext);
console.log(dates)

const MILLISECONDS_PER_DAY = 1000 * 60 * 60 * 24;
function dayDifference(date1, date2) {
  const timeDiff = Math.abs(date2.getTime() - date1.getTime());
  const diffDays = Math.ceil(timeDiff / MILLISECONDS_PER_DAY);
  return diffDays;
}

const days = dayDifference(dates[0].endDate, dates[0].startDate);

const handleOpen = (i) => {
  setSlideNumber(i);
  setOpen(true);
};

const handleMove = (direction) => {
  let newSlideNumber;

  if (direction === "l") {
    newSlideNumber = slideNumber === 0 ? 5 : slideNumber - 1;
  } else {
    newSlideNumber = slideNumber === 5 ? 0 : slideNumber + 1;
  }

  setSlideNumber(newSlideNumber);
};

const handleClick = () => {
  if (user) {
    setOpenModal(true);
  } else {
    setOpenModal(true);
```

```
  }
};

return (
  <div>
    <Navbar />
    <Header type="list" />
    {loading ? (
      "loading"
    ) : (
      <div className="hotelContainer">
        {open && (
          <div className="slider">
            <FontAwesomeIcon
              icon={faCircleXmark}
              className="close"
              onClick={() => setOpen(false)}
            />
            <FontAwesomeIcon
              icon={faCircleArrowLeft}
              className="arrow"
              onClick={() => handleMove("l")}
            />
            <div className="sliderWrapper">
              <img
                src={data.photos[slideNumber]}
                alt=""
                className="sliderImg"
              />
            </div>
            <FontAwesomeIcon
              icon={faCircleArrowRight}
              className="arrow"
              onClick={() => handleMove("r")}
            />
          </div>
        )}
        <div className="hotelWrapper">
          <button className="bookNow">Reserve or Book Now!</button>
          <h1 className="hotelTitle">{data.name}</h1>
          <div className="hotelAddress">
            <FontAwesomeIcon icon={faLocationDot} />
            <span>{data.address}</span>
```

```
          </div>
          <span className="hotelDistance">
            Excellent location - {data.distance}m from center
          </span>
          <span className="hotelPriceHighlight">
            Book a stay over ${data.cheapestPrice} at this property and get a
            free airport taxi
          </span>
          <div className="hotelImages">
            {data.photos?.map((photo, i) => (
              <div className="hotelImgWrapper" key={i}>
                <img
                  onClick={() => handleOpen(i)}
                  src={photo}
                  alt=""
                  className="hotelImg"
                />
              </div>
            ))}
          </div>
          <div className="hotelDetails">
            <div className="hotelDetailsTexts">
              <h1 className="hotelTitle">{data.title}</h1>
              <p className="hotelDesc">{data.desc}</p>
            </div>
            <div className="hotelDetailsPrice">
              <h1>Perfect for a {days}-night stay!</h1>
              <span>
                Located in the real heart of Krakow, this property has an
                excellent location score of 9.8!
              </span>
              <h2>
                <b>${days * data.cheapestPrice * options.room}</b> ({days}{" "}
                nights)
              </h2>
              <button onClick={handleClick}>Reserve or Book Now!</button>
            </div>
          </div>
        </div>
        <MailList />
        <Footer />
      </div>
    )}
```

```
        {openModal && <Reserve setOpen={setOpenModal} hotelId={id}/>}
    </div>
  );
};

export default Hotel;
```

client/

```
import "./list.css";
import Navbar from "../../components/navbar/Navbar";
import Header from "../../components/header/Header";
import { useLocation } from "react-router-dom";
import { useState } from "react";
import { format, setDate } from "date-fns";
import { DateRange } from "react-date-range";
import SearchItem from "../../components/searchItem/SearchItem";
import useFetch from "../../hooks/useFetch";
import reFetch from "../../hooks/useFetch";



const List = () => {
  const location = useLocation();
  const [destination, setDestination] = useState(location.state.destination);
  const [dates, setDates] = useState(location.state.dates);
  const [openDate, setOpenDate] = useState(false);
  const [options, setOptions] = useState(location.state.options);
  const [min, setMin] = useState(undefined);

  const [max, setMax] = useState(undefined);

  const {data, loading, error, reFetch} =
useFetch(`/hotels?city=${destination}&min=${min || 0}&max=${max || 999}`);

  const handleClick = () => {
    reFetch()
  };



  return (
```

```
    <div>
      <Navbar />
      <Header type="list" />
      <div className="listContainer">
        <div className="listWrapper">
          <div className="listSearch">
            <h1 className="lsTitle">Search</h1>
            <div className="lsItem">
              <label>Destination</label>
              <input placeholder={destination} type="text" />
            </div>
            <div className="lsItem">
              <label>Check-in Date</label>
              <span onClick={() => setOpenDate(!openDate)}>{`${format(
                dates[0].startDate,
                "MM/dd/yyyy"
              )} to ${format(dates[0].endDate, "MM/dd/yyyy")}`}</span>
              {openDate && (
                <DateRange
                  onChange={(item) => setDate([item.selection])}
                  minDate={new Date()}
                  ranges={dates}
                />
              )}
            </div>
            <div className="lsItem">
              <label>Options</label>
              <div className="lsOptions">
                <div className="lsOptionItem">
                  <span className="lsOptionText">
                    Min price <small>per night</small>
                  </span>
                  <input type="number" onChange={e=>setMin(e.target.value)}
className="lsOptionInput" />
                </div>
                <div className="lsOptionItem">
                  <span className="lsOptionText">
                    Max price <small>per night</small>
                  </span>
                  <input type="number" onChange={e=>setMax(e.target.value)}
className="lsOptionInput" />
                </div>
                <div className="lsOptionItem">
```

```
                <span className="lsOptionText">Adult</span>
                <input
                  type="number"
                  min={1}
                  className="lsOptionInput"
                  placeholder={options.adult}
                />
              </div>
              <div className="lsOptionItem">
                <span className="lsOptionText">Children</span>
                <input
                  type="number"
                  min={0}
                  className="lsOptionInput"
                  placeholder={options.children}
                />
              </div>
              <div className="lsOptionItem">
                <span className="lsOptionText">Room</span>
                <input
                  type="number"
                  min={1}
                  className="lsOptionInput"
                  placeholder={options.room}
                />
              </div>
            </div>
          </div>
          <button onClick={handleClick}>Search</button>
        </div>
        <div className="listResult">
          {loading ? "loading": <>
          {data.map (item=>(
            <SearchItem item = {item} key={item._id}/>
          ))}
          </>}

        </div>
      </div>
    </div>
  </div>
  );
};
```

```
export default List;
```

client/pages/login.js

```
import axios from "axios";
import { useContext, useState } from "react";
import { useNavigate } from "react-router-dom";
import { AuthContext } from "../../context/AuthContext";
import "./login.css";

const Login = () => {
  const [credentials, setCredentials] = useState({
    username: undefined,
    password: undefined,
  });

  const {loading, error, dispatch } = useContext(AuthContext);

  const navigate = useNavigate()

  const handleChange = (e) => {
    setCredentials((prev) => ({ ...prev, [e.target.id]: e.target.value }));
  };

  const handleClick = async (e) => {
    e.preventDefault();
    dispatch({ type: "LOGIN_START" });
    try {
      const res = await axios.post("/auth/login", credentials);
      dispatch({ type: "LOGIN_SUCCESS", payload: res.data.details });
      navigate("/")
    } catch (err) {
      dispatch({ type: "LOGIN_FAILURE", payload: err.response.data });
    }
  };



  return (
    <div className="login">
      <div className="lContainer">
        <input
          type="text"
```

```
          placeholder="username"
          id="username"
          onChange={handleChange}
          className="lInput"
        />
        <input
          type="password"
          placeholder="password"
          id="password"
          onChange={handleChange}
          className="lInput"
        />
        <button disabled={loading} onClick={handleClick} className="lButton">
          Login
        </button>
        {error && <span>{error.message}</span>}
      </div>
    </div>
  );
};

export default Login;
```

AuthContext:

```
import { createContext, useEffect, useReducer } from "react";

const INITIAL_STATE = {
  user: JSON.parse(localStorage.getItem("user")) || null,
  loading: false,
  error: null,
};

export const AuthContext = createContext(INITIAL_STATE);

const AuthReducer = (state, action) => {
  switch (action.type) {
    case "LOGIN_START":
      return {
        user: null,
        loading: true,
        error: null,
      };
```

```
    case "LOGIN_SUCCESS":
      return {
        user: action.payload,
        loading: false,
        error: null,
      };
    case "LOGIN_FAILURE":
      return {
        user: null,
        loading: false,
        error: action.payload,
      };
    case "LOGOUT":
      return {
        user: null,
        loading: false,
        error: null,
      };
    default:
      return state;
  }
};

export const AuthContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AuthReducer, INITIAL_STATE);

  useEffect(() => {
    localStorage.setItem("user", JSON.stringify(state.user));
  }, [state.user]);

  return (
    <AuthContext.Provider
      value={{
        user: state.user,
        loading: state.loading,
        error: state.error,
        dispatch,
      }}
    >
      {children}
    </AuthContext.Provider>
  );
};
```

SearchContext:

```
import { createContext, useEffect, useReducer } from "react";

const INITIAL_STATE = {
  user: JSON.parse(localStorage.getItem("user")) || null,
  loading: false,
  error: null,
};

export const AuthContext = createContext(INITIAL_STATE);

const AuthReducer = (state, action) => {
  switch (action.type) {
    case "LOGIN_START":
      return {
        user: null,
        loading: true,
        error: null,
      };
    case "LOGIN_SUCCESS":
      return {
        user: action.payload,
        loading: false,
        error: null,
      };
    case "LOGIN_FAILURE":
      return {
        user: null,
        loading: false,
        error: action.payload,
      };
    case "LOGOUT":
      return {
        user: null,
        loading: false,
        error: null,
      };
    default:
      return state;
  }
};
```

```
export const AuthContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(AuthReducer, INITIAL_STATE);

  useEffect(() => {
    localStorage.setItem("user", JSON.stringify(state.user));
  }, [state.user]);

  return (
    <AuthContext.Provider
      value={{
        user: state.user,
        loading: state.loading,
        error: state.error,
        dispatch,
      }}
    >
      {children}
    </AuthContext.Provider>
  );
};
```

admin/app.js

```
import Home from "./pages/home/Home";
import Login from "./pages/login/Login";
import List from "./pages/list/List";
import Single from "./pages/single/Single";
import New from "./pages/new/New";
import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom";
import { productInputs, userInputs } from "./formSource";
import "./style/dark.scss";
import { useContext } from "react";
import { DarkModeContext } from "./context/darkModeContext";
import { AuthContext } from "./context/AuthContext";
import { hotelColumns, roomColumns, userColumns } from "./datatablesource";
import NewHotel from "./pages/newHotel/NewHotel";
import NewRoom from "./pages/newRoom/NewRoom";

function App() {
  const { darkMode } = useContext(DarkModeContext);

  const ProtectedRoute = ({ children }) => {
    const { user } = useContext(AuthContext);
```

```
  if (!user) {
    return <Navigate to="/login" />;
  }

  return children;
};

return (
  <div className={darkMode ? "app dark" : "app"}>
    <BrowserRouter>
      <Routes>
        <Route path="/">
          <Route path="login" element={<Login />} />
          <Route
            index
            element={
              <ProtectedRoute>
                <Home />
              </ProtectedRoute>
            }
          />
          <Route path="users">
            <Route
              index
              element={
                <ProtectedRoute>
                  <List columns={userColumns} />
                </ProtectedRoute>
              }
            />
            <Route
              path=":userId"
              element={
                <ProtectedRoute>
                  <Single />
                </ProtectedRoute>
              }
            />
            <Route
              path="new"
              element={
                <ProtectedRoute>
```

```
              <New inputs={userInputs} title="Add New User" />
            </ProtectedRoute>
          }
        />
      </Route>
      <Route path="hotels">
        <Route
          index
          element={
            <ProtectedRoute>
              <List columns={hotelColumns} />
            </ProtectedRoute>
          }
        />
        <Route
          path=":productId"
          element={
            <ProtectedRoute>
              <Single />
            </ProtectedRoute>
          }
        />
        <Route
          path="new"
          element={
            <ProtectedRoute>
              <NewHotel  />
            </ProtectedRoute>
          }
        />
      </Route>
      <Route path="rooms">
        <Route
          index
          element={
            <ProtectedRoute>
              <List columns={roomColumns} />

            </ProtectedRoute>
          }
        />
        <Route
          path=":productId"
```

```
                element={
                  <ProtectedRoute>
                    <Single />
                  </ProtectedRoute>
                }
              />
              <Route
                path="new"
                element={
                  <ProtectedRoute>
                    <NewRoom  />
                  </ProtectedRoute>
                }
              />
            </Route>
          </Route>
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

admin/datatablesource.js

```
export const userColumns = [
  { field: "id", headerName: "ID", width: 70 },
  {
    field: "user",
    headerName: "User",
    width: 230,
    renderCell: (params) => {
      return (
        <div className="cellWithImg">
          <img className="cellImg" src={params.row.img ||
"https://i.ibb.co/MBtjqXQ/no-avatar.gif"} alt="avatar" />
          {params.row.username}
        </div>
      );
    },
  },
  {
```

```
    field: "email",
    headerName: "Email",
    width: 230,
  },

  {
    field: "country",
    headerName: "Country",
    width: 100,
  },
  {
    field: "city",
    headerName: "City",
    width: 100,
  },
  {
    field: "phone",
    headerName: "Phone",
    width: 100,
  },

];

export const hotelColumns = [
  { field: "_id", headerName: "ID", width: 250 },
  {
    field: "name",
    headerName: "Name",
    width: 150,
  },
  {
    field: "type",
    headerName: "Type",
    width: 100,
  },
  {
    field: "title",
    headerName: "Title",
    width: 230,
  },
  {
    field: "city",
    headerName: "City",
```

```
    width: 100,
  },
];

export const roomColumns = [
  { field: "_id", headerName: "ID", width: 70 },
  {
    field: "title",
    headerName: "Title",
    width: 230,
  },
  {
    field: "desc",
    headerName: "Description",
    width: 200,
  },
  {
    field: "price",
    headerName: "Price",
    width: 100,
  },
  {
    field: "maxPeople",
    headerName: "Max People",
    width: 100,
  },
];
```

admin/index.js

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";
import { DarkModeContextProvider } from "./context/darkModeContext";
import {AuthContextProvider} from "./context/AuthContext"


ReactDOM.render(
  <React.StrictMode>
    <AuthContextProvider>
    <DarkModeContextProvider>
      <App />
    </DarkModeContextProvider></AuthContextProvider>
```

```
  </React.StrictMode>,
  document.getElementById("root")
);
```

Admin/newHotel.js

```
import "./newHotel.scss";
import Sidebar from "../../components/sidebar/Sidebar";
import Navbar from "../../components/navbar/Navbar";
import DriveFolderUploadOutlinedIcon from
"@mui/icons-material/DriveFolderUploadOutlined";
import { useState } from "react";
import { hotelInputs } from "../../formSource";
import useFetch from "../../hooks/useFetch";
import axios from "axios";

const NewHotel = () => {
  const [files, setFiles] = useState("");
  const [info, setInfo] = useState({});
  const [rooms, setRooms] = useState([]);

  const { data, loading, error } = useFetch("/rooms");

  const handleChange = (e) => {
    setInfo((prev) => ({ ...prev, [e.target.id]: e.target.value }));
  };

  const handleSelect = (e) => {
    const value = Array.from(
      e.target.selectedOptions,
      (option) => option.value
    );
    setRooms(value);
  };

  console.log(files)

  const handleClick = async (e) => {
    e.preventDefault();
    try {
      const list = await Promise.all(
        Object.values(files).map(async (file) => {
```

```
          const data = new FormData();
          data.append("file", file);
          data.append("upload_preset", "upload");
          const uploadRes = await axios.post(
            "https://api.cloudinary.com/v1_1/di0tmfdnr/image/upload",
            data
          );

          const { url } = uploadRes.data;
          return url;
        })
      );

      const newhotel = {
        ...info,
        rooms,
        photos: list,
      };

      await axios.post("/hotels", newhotel);
    } catch (err) {console.log(err)}
  };
  return (
    <div className="new">
      <Sidebar />
      <div className="newContainer">
        <Navbar />
        <div className="top">
          <h1>Add New Product</h1>
        </div>
        <div className="bottom">
          <div className="left">
            <img
              src={
                files
                  ? URL.createObjectURL(files[0])
                  :
"https://icon-library.com/images/no-image-icon/no-image-icon-0.jpg"
              }
              alt=""
            />
          </div>
          <div className="right">
```

```
          <form>
            <div className="formInput">
              <label htmlFor="file">
                Image: <DriveFolderUploadOutlinedIcon className="icon" />
              </label>
              <input
                type="file"
                id="file"
                multiple
                onChange={(e) => setFiles(e.target.files)}
                style={{ display: "none" }}
              />
            </div>

            {hotelInputs.map((input) => (
              <div className="formInput" key={input.id}>
                <label>{input.label}</label>
                <input
                  id={input.id}
                  onChange={handleChange}
                  type={input.type}
                  placeholder={input.placeholder}
                />
              </div>
            ))}
            <div className="formInput">
              <label>Featured</label>
              <select id="featured" onChange={handleChange}>
                <option value={false}>No</option>
                <option value={true}>Yes</option>
              </select>
            </div>
            <div className="selectRooms">
              <label>Rooms</label>
              <select id="rooms" multiple onChange={handleSelect}>
                {loading
                  ? "loading"
                  : data &&
                    data.map((room) => (
                      <option key={room._id} value={room._id}>
                        {room.title}
                      </option>
                    ))}
```

```
                </select>
              </div>
              <button onClick={handleClick}>Send</button>
            </form>
          </div>
        </div>
      </div>
    </div>
  );
};

export default NewHotel;
```

Admin/newRoom.js

```
import "./newRoom.scss";
import Sidebar from "../../components/sidebar/Sidebar";
import Navbar from "../../components/navbar/Navbar";
import DriveFolderUploadOutlinedIcon from
"@mui/icons-material/DriveFolderUploadOutlined";
import { useState } from "react";
import { roomInputs } from "../../formSource";
import useFetch from "../../hooks/useFetch";
import axios from "axios";

const NewRoom = () => {
  const [info, setInfo] = useState({});
  const [hotelId, setHotelId] = useState(undefined);
  const [rooms, setRooms] = useState([]);

  const { data, loading, error } = useFetch("/hotels");

  const handleChange = (e) => {
    setInfo((prev) => ({ ...prev, [e.target.id]: e.target.value }));
  };

  const handleClick = async (e) => {
    e.preventDefault();
    const roomNumbers = rooms.split(",").map((room) => ({ number: room }));
    try {
      await axios.post(`/rooms/${hotelId}`, { ...info, roomNumbers });
    } catch (err) {
      console.log(err);
```

```
  }
};

console.log(info)
return (
  <div className="new">
    <Sidebar />
    <div className="newContainer">
      <Navbar />
      <div className="top">
        <h1>Add New Room</h1>
      </div>
      <div className="bottom">
        <div className="right">
          <form>
            {roomInputs.map((input) => (
              <div className="formInput" key={input.id}>
                <label>{input.label}</label>
                <input
                  id={input.id}
                  type={input.type}
                  placeholder={input.placeholder}
                  onChange={handleChange}
                />
              </div>
            ))}
            <div className="formInput">
              <label>Rooms</label>
              <textarea
                onChange={(e) => setRooms(e.target.value)}
                placeholder="give comma between room numbers."
              />
            </div>
            <div className="formInput">
              <label>Choose a hotel</label>
              <select
                id="hotelId"
                onChange={(e) => setHotelId(e.target.value)}
              >
                {loading
                  ? "loading"
                  : data &&
                    data.map((hotel) => (
```

```
                        <option key={hotel._id}
value={hotel._id}>{hotel.name}</option>
                      ))}
                  </select>
                </div>
                <button onClick={handleClick}>Send</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    );
};

export default NewRoom;
```

Admin/Home.jsx

```
import Sidebar from "../../components/sidebar/Sidebar";
import Navbar from "../../components/navbar/Navbar";
import "./home.scss";
import Widget from "../../components/widget/Widget";
import Featured from "../../components/featured/Featured";
import Chart from "../../components/chart/Chart";
import Table from "../../components/table/Table";

const Home = () => {
  return (
    <div className="home">
      <Sidebar />
      <div className="homeContainer">
        <Navbar />
        <div className="widgets">
          <Widget type="user" />
          <Widget type="order" />
          <Widget type="earning" />
          <Widget type="balance" />
        </div>
        <div className="charts">
          <Featured />
          <Chart title="Last 6 Months (Revenue)" aspect={2 / 1} />
        </div>
        <div className="listContainer">
```

```
          <div className="listTitle">Latest Transactions</div>
          <Table />
        </div>
      </div>
    </div>
  );
};

export default Home;
```

For all Source Codes Please see the attached zip file.