A Project Report on

QUICK BOOK HOTEL BOOKING

Submitted by

S Sarfraj – R170317

L Umar Ahmad – R170930

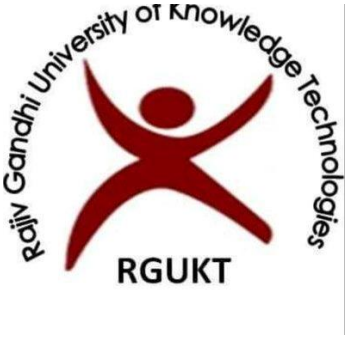
R L Rajasekhar Reddy – R170129

Submitted to

IIIT RK VALLEY

Idupulapaya, Vempalli, YSR Kadapa

Andhra Pradesh, India PIN 516330



Under the guidance of

M HimaBindu

Assistant Professor

As a part of

Partial fulfilment of the degree of Bachelor of Technology in

Computer Science and Engineering

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

R.K. Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is certify that the project entitled "**QUICK BOOK HOTEL BOOKING"** submitted by **S.Sarfraj** (**R170317**), **L.Umar Ahmad** (**R170930**), **R.L.Rajeskhar Reddy** (**R170129**),under our guidance and supervision for the partial fulfilment for degree Bachelor of Technology in Computer Science and Engineering – 4 during the academic semester – I and semester – II 2022-2023 at RGUKT, RK VALLEY. To the best of our knowledge, the report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma

Project Internal Guide

M HimaBindu,

Assistant Professor

Computer Science and Engineering

RK Valley, RGUKT.

Head of the Department

N Satyanandaram

Assistant Professor

Computer Science and Engineering

RK Valley, RGUKT.

Submitted for the practical examination held on …….

Internal Examiner

External Examiner

Acknowledgement

We would like to express our sincere gratitude to **Ms. M HimaBindu** Mam, our project internal guide for valuable suggestions and keen interest throughout the progress of my course of research.

We are grateful to **Mr. N Satyanandaram** sir, HOD CSE, for providing excellent computing facilities and a congenial atmosphere for progressing with our project.

At the outset, we would like to thank **Rajiv Gandhi University of Knowledge Technologies** for providing all the necessary resources for the successful completion of our course work.

Index

S.no	Title	Page no
1	Abstract	5
2	Introduction	6 – 7
3	Technologies	7 - 8
4	Software Configurations	8
5	Design	9
6	Use Case Diagram	10
7	ER Diagram	11
8	Coding	12 – 21
9	Testing	22
10	Future Improvements	23
11	Snippets	24
12	References	25

Abstract

The Quick Book Hotel Booking System is a project implemented for Hotel Reservation, which is an Online Hotel Booking System. It provides people all Over the world with an easy and fast way to book hotel rooms online. The interface of the Quick Book Hotel Booking System is Web pages that can be accessed with a Web site browser. The system is implemented in MERN(Mongo DB, Express JS, React JS, Node JS). Users can perform room booking activities at Hotel anytime and anywhere by accessing it via Internet. The Online Hotel Booking System is an easy-to-use application. Everyone who knows how to use a Web browser can easily carry out booking, change the booking details, cancel the booking, change the personal profile, view the booking history, or view the hotel information by following its simple and clear GUI (Graphical user interface) design.

Introduction

The goal of this project is to create a user-friendly web application that allows customers to browse and book hotels online. Our application will provide a comprehensive listing of hotels in various locations, with information on availability, rates, amenities, and user reviews.

1.1: Purpose

The app would allow users to browse through a list of hotels, filter results by price range, location, amenities, and other factors, and book their preferred hotel room directly through the app.

1.2: Intended Audience:

The intended audience will be the user, were he can book the hotel rooms during trips and can have a lot of options to book rooms.

Product Vision:

Quick book hotel booking creates a platform that provides users with an easy and stress-free experience when booking their hotel accommodations, while also providing value for hotels through increased bookings and revenue.

Technologies:

➢ HTML

➢CSS

➢Java script

➢Node JS

➢Express JS

➢ MongoDB

Reactjs :

React is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta and a community of individual developers and companies. React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js.

Node JS :

Node JS is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Mongo DB :

Mongo DB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in Mongo DB. Collections contain sets of documents and function which is the equivalent of relational database tables.

Collections → Table

Documents → Rows

Express JS :

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application.

It's a layer built on the top of the Node js that helps manage servers and routes

Express was created to make APIs and web applications with ease,

It saves a lot of coding time almost by half and still makes web and mobile applications are efficient.

Another reason for using express is that it is written in JavaScript as java script is an easy language

even if you don't have a previous knowledge of any language. Express lets so many new developers enter the field of web development.

The reason behind creating an express framework for Node JS is

➢Time Efficient

➢Fast

➢Economical

➢Easy to learn

➢Asynchronous

Software Configurations

➢Node.js v16.17.0

➢Ubuntu 18.04 LTS

Design

Modules

1.Login

2.Registration

3.Home

4.Hotel Review

5.Reservation

Login & Registration:

In this module if the user is new user, then he has to create an account by giving the user credentials according to the in-registration form

If the user has an already account, then user have to login with his/her Email id and password

Home:

In this module all hotel site of all areas details are be displayed and you can search the hotels by their names

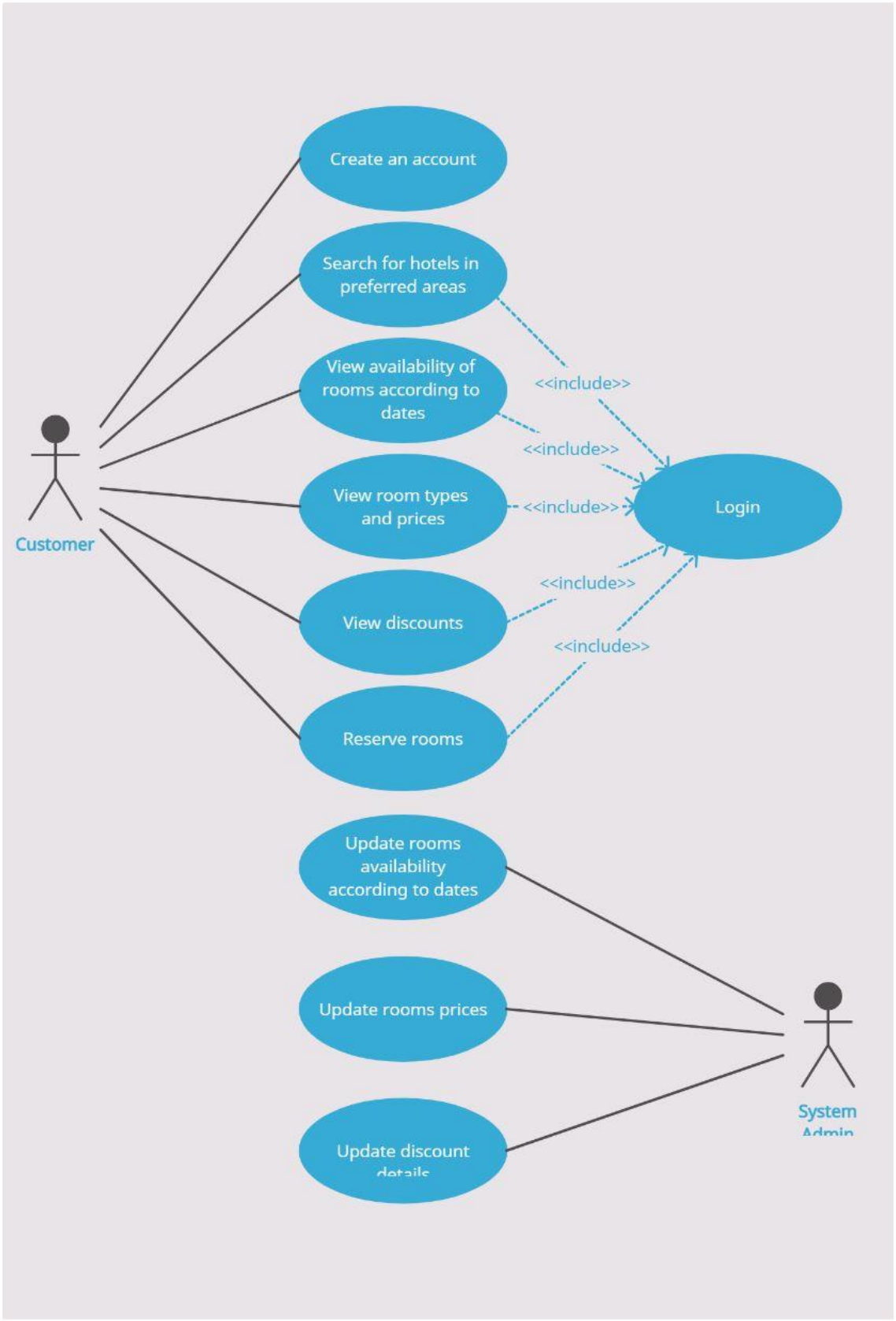
Hotel Review:

In this module the user can view the details of the selected hotel like Building, Rooms etc.

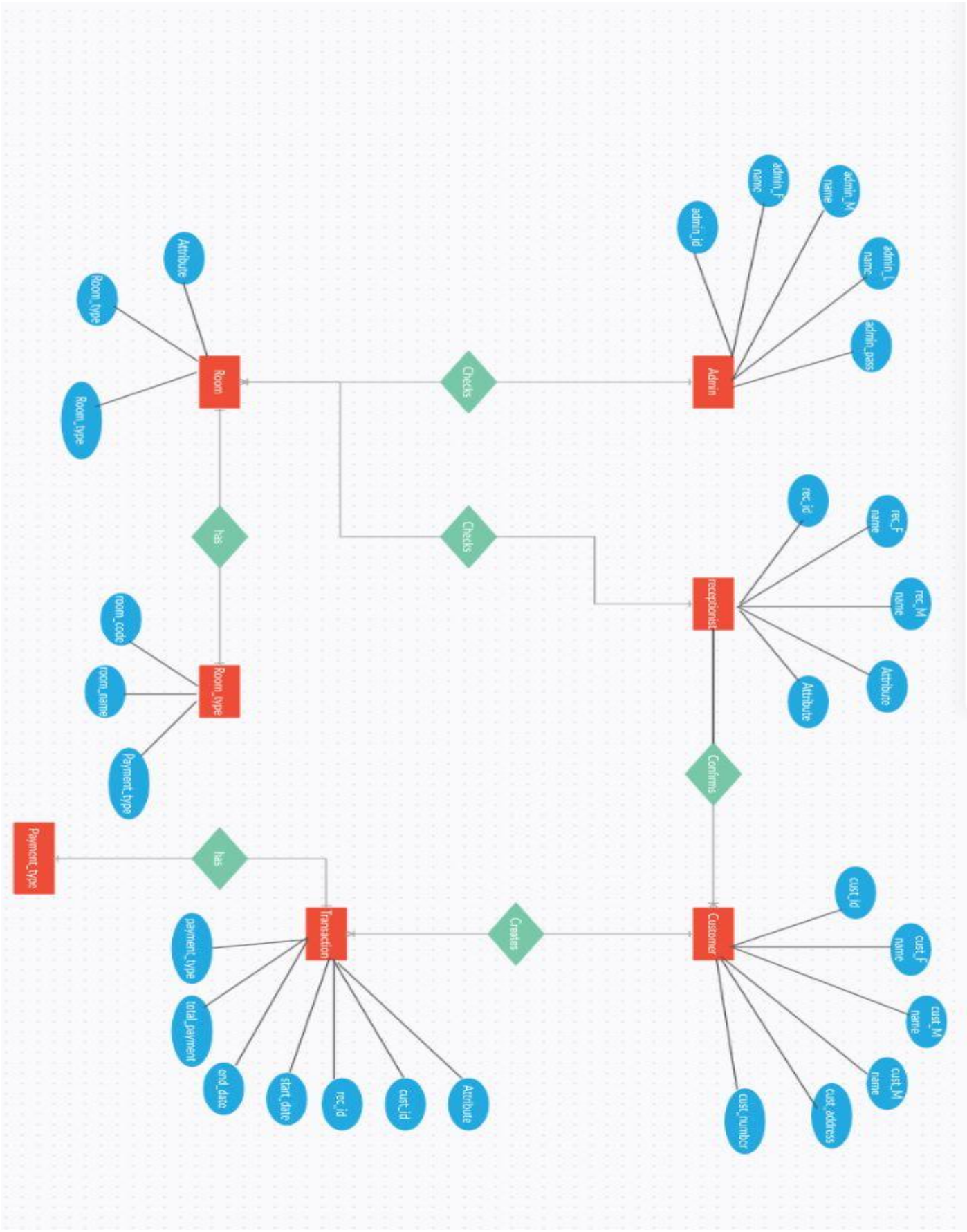
Reservation:

In this module all of the user can see their reservation details

Use Case Diagram :



ER Diagram :


Attribute
Room_type
Room
Room_type
Checks
admin_id
admin_F name
admin_M name
admin_L name
admin_pass
Admin
has
Checks
rec_id
rec_F name
rec_M name
Attribute
Attribute
receptionist
room_code
room_name
Room_type
Payment_type
Confirms
Payment_type
has
payment_type
total_payment
cmd_date
start_date
rec_id
cust_id
Attribute
Transaction
Creates
Customer
cust_id
cust_F name
cust_M name
cust_M name
cust_address
cust_number

Coding

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>
);
```

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';
import Register from './pages/register/Register';
```

```
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/>} />
				<Route path="/register" element={<Register />} />
			</Routes>
		</BrowserRouter>
		);
}

export default App;
```

Header.js:

```
import React, { useContext, useState } from 'react'
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import {
	faBed,
	faCalendarDays,
	faCar,
	faPerson,
	faPlane,
	faTaxi,
} from "@fortawesome/free-solid-svg-icons";import './header.css'
import { DateRange } from 'react-date-range'
import 'react-date-range/dist/styles.css'; // main css file
```

```
import 'react-date-range/dist/theme/default.css'; // theme css file
import { format } from 'date-fns';
import { useNavigate } from 'react-router-dom'
import { SearchContext } from '../../context/SearchContext';
import { AuthContext } from '../../context/AuthContext';

const Header = ({ type }) => {
    const [openDate, setOpenDate] = useState(false);
    const [destination, setDestination] = useState("");
    const [dates, setDates] = useState([
      {
        startDate: new Date(),
        endDate: new Date(),
        key: 'selection'
      }
    ]);
    const [openOptions, setOpenOptions] = useState(false);
    const [options, setOptions] = useState({
      adult: 1,
      children: 0,
      room: 1,
    });

    const navigate = useNavigate();
    const { user } = useContext(AuthContext);

    const handleOption = (name, operation) => {
      setOptions((prev) => {
        return {
          ...prev,
          [name]: operation === "i" ? options[name] +1 : options[name] -1,
        };
      });
```

```
};

const { dispatch } = useContext(SearchContext);

const handleSearch = () => {
    dispatch({ type: "NEW_SEARCH", payload: { destination, dates, options } })
    navigate("/hotels", {state: { destination, dates, options }})
}

const handleLogin = () => {
  navigate("/login");
}
      return (
                <div className='header'>
  <div className={type === "list" ? "headerContainer listMode": "headerContainer"}>
                        <div className="headerList">
      <div className="headerListItem active">
        <FontAwesomeIcon icon={faBed} />
        <span>Stays</span>
      </div>
      <div className="headerListItem">
        <FontAwesomeIcon icon={faPlane} />
        <span>Flights</span>
      </div>
      <div className="headerListItem">
        <FontAwesomeIcon icon={faCar} />
        <span>Car rentals</span>
      </div>
      <div className="headerListItem">
        <FontAwesomeIcon icon={faBed} />
        <span>Attractions</span>
      </div>
      <div className="headerListItem">
```

```
            <FontAwesomeIcon icon={faTaxi} />
            <span>Airport taxis</span>
          </div>
        </div>
                    { type !== "list" && <><h1 className="headerTitle">
          A lifetime of discounts? It's Genius.
        </h1>
        <p className="headerDesc">
          Get rewarded for your travels – unlock instant savings of 10% or
          more with a free booking account
        </p>
        {!user && <button onClick={handleLogin} className="headerBtn">Sign in / Register</button>}
        <div className='headerSearch'>
          <div className='headerSearchItem'>
            <FontAwesomeIcon icon={faBed} className='headerIcon' />
            <input type='text' placeholder='where are you going?' className='headerSearchInput' onChange={e => setDestination(e.target.value)} />
          </div>
          <div className='headerSearchItem'>
            <FontAwesomeIcon icon={faCalendarDays} className='headerIcon' />
            <span onClick={() => setOpenDate(!openDate)} className='headerSearchText'>{`${format(dates[0].startDate, "MM/dd/yyyy")} to ${format(dates[0].endDate, "MM/dd/yyyy")}`}</span>
            {openDate && <DateRange
              editableDateInputs={true}
              onChange={ item => setDates([item.selection])}
              moveRangeOnFirstSelection={false}
              ranges={dates}
              className='date'
              minDate={new Date() }
            />}
          </div>
          <div className='headerSearchItem'>
            <FontAwesomeIcon icon={faPerson} className='headerIcon' />
```

```
<span onClick={() => setOpenOptions(!openOptions)} className='headerSearchText'>{`${options.adult} adult . ${options.children} children . ${options.room} room `}</span>

{openOptions && <div className='options'>

<div className='optionsItem'>

<span className='optionText'>Adult</span>

<div className='optionCounter'>

<button disabled={options.adult <= 1} className='optionCounterButton' onClick={() => handleOption("adult", "d")}>-</button>

<span className='optionCounterNumber'>{options.adult}</span>

<button className='optionCounterButton' onClick={() => handleOption("adult", "i")}>+</button>

</div>

</div>

<div className='optionsItem'>

<span className='optionText'>Children</span>

<div className='optionCounter'>

<button disabled={options.children <= 0} className='optionCounterButton' onClick={() => handleOption("children", "d")}>-</button>

<span className='optionCounterNumber'>{options.children}</span>

<button className='optionCounterButton' onClick={() => handleOption("children", "i")}>+</button>

</div>

</div>

<div className='optionsItem'>

<span className='optionText'>Rooms</span>

<div className='optionCounter'>

<button disabled={options.room <= 1} className='optionCounterButton' onClick={() => handleOption("room", "d")}>-</button>

<span className='optionCounterNumber'>{options.room}</span>

<button className='optionCounterButton' onClick={() => handleOption("room", "i")}>+</button>

</div>

</div>


</div>
```

```
          }
        </div>
        <div className='headerSearchItem'>
          <button className='headerBtn' onClick={handleSearch}>Search</button>
        </div>
       </div>
       </>
       }
      </div>
      </div>
        );
}

export default Header
```

Reserve.js:

```
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faCircleXmark } from "@fortawesome/free-solid-svg-icons";

import "./reserve.css";
import useFetch from "../../hooks/useFetch";
import { useContext, useState } from "react";
import { SearchContext } from "../../context/SearchContext";
import axios from "axios";
import { useNavigate } from "react-router-dom";

const Reserve = ({ setOpen, hotelId }) => {
  const [selectedRooms, setSelectedRooms] = useState([]);
  const { data, loading, error } = useFetch(`/hotels/room/${hotelId}`);
  const { dates } = useContext(SearchContext);
```

```
const getDatesInRange = (startDate, endDate) => {
  const start = new Date(startDate);
  const end = new Date(endDate);

  const date = new Date(start.getTime());

  const dates = [];

  while (date <= end) {
    dates.push(new Date(date).getTime());
    date.setDate(date.getDate() + 1);
  }

  return dates;
};

const alldates = getDatesInRange(dates[0].startDate, dates[0].endDate);

const isAvailable = (roomNumber) => {
  const isFound = roomNumber.unavailableDates.some((date) =>
    alldates.includes(new Date(date).getTime())
  );

  return !isFound;
};

const handleSelect = (e) => {
  const checked = e.target.checked;
  const value = e.target.value;
  setSelectedRooms(
    checked
      ? [...selectedRooms, value]
      : selectedRooms.filter((item) => item !== value)
```

```
  );
};

const navigate = useNavigate();

const handleClick = async () => {
  try {
    await Promise.all(
      selectedRooms.map((roomId) => {
        const res = axios.put(`/rooms/availability/${roomId}`, {
          dates: alldates,
        });
        return res.data;
      })
    );
    setOpen(false);
    navigate("/");
  } catch (err) {}
};
return (
  <div className="reserve">
    <div className="rContainer">
      <FontAwesomeIcon
        icon={faCircleXmark}
        className="rClose"
        onClick={() => setOpen(false)}
      />
      <span>Select your rooms:</span>
      {data.map((item) => (
        <div className="rItem" key={item._id}>
          <div className="rItemInfo">
            <div className="rTitle">{item.title}</div>
            <div className="rDesc">{item.desc}</div>
```

```
            <div className="rMax">
              Max people: <b>{item.maxPeople}</b>
            </div>
            <div className="rPrice">{item.price}</div>
          </div>
          <div className="rSelectRooms">
            {item.roomNumbers.map((roomNumber) => (
              <div className="room">
                <label>{roomNumber.number}</label>
                <input
                  type="checkbox"
                  value={roomNumber._id}
                  onChange={handleSelect}
                  disabled={!isAvailable(roomNumber)}
                />
              </div>
            ))}
          </div>
        </div>
      ))}
      <button onClick={handleClick} className="rButton">
        Reserve Now!
      </button>
    </div>
  </div>
 );
};

export default Reserve;
```

Testing

Here we performed two types of testing to the software for finding bugs

1.Functional Testing:

we tested main features like testing each and every module like login ,signup, Profile, Hotels, Hotel Search,Booking.

Integration Testing:

Here, the data flow is tested .For example ,if we take login module by entering valid credentials it redirects to the respected user's Dashboard .

System Testing:

Here, the end to end Testing is done on application from entering credentials, navigating to the all modules such as Profile of the User etc. and at last to the logout page.

2.Non-Functional Testing:

Here we tested the Non-functional features like Compatibility, Performance

Compatibility Testing:

Here We tested this software on Various Operating System Such as Linux, windows etc…

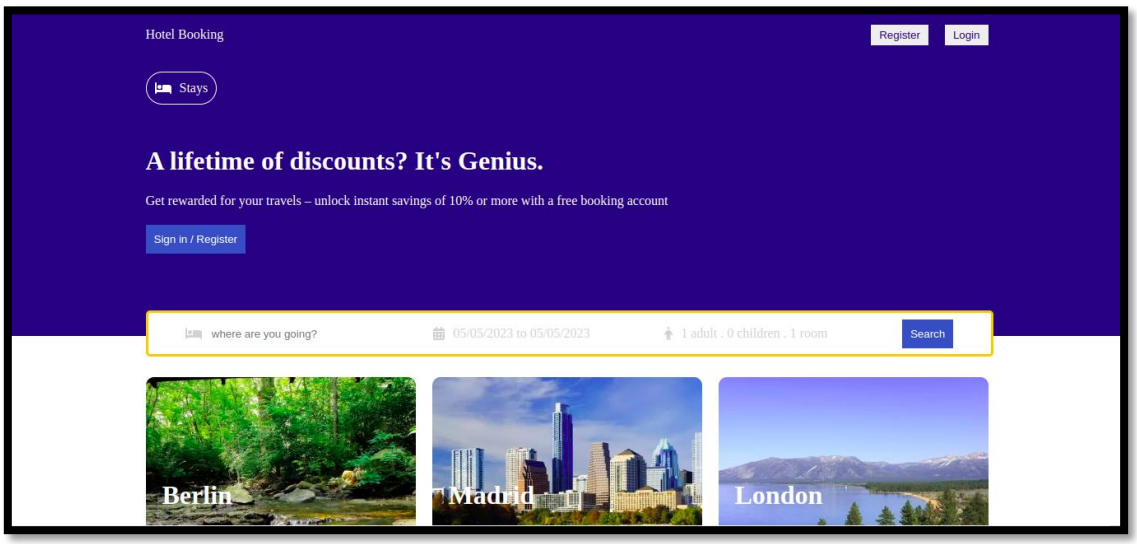
Performance Testing

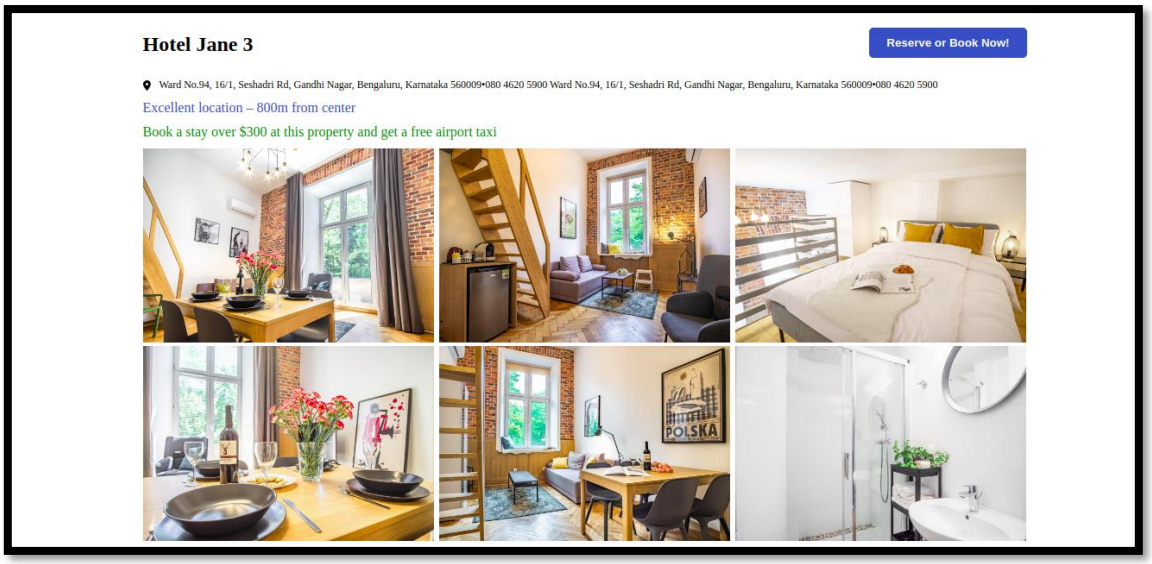
Here we tested the speed, efficiency. The software is given accurate results when the user enters the data.

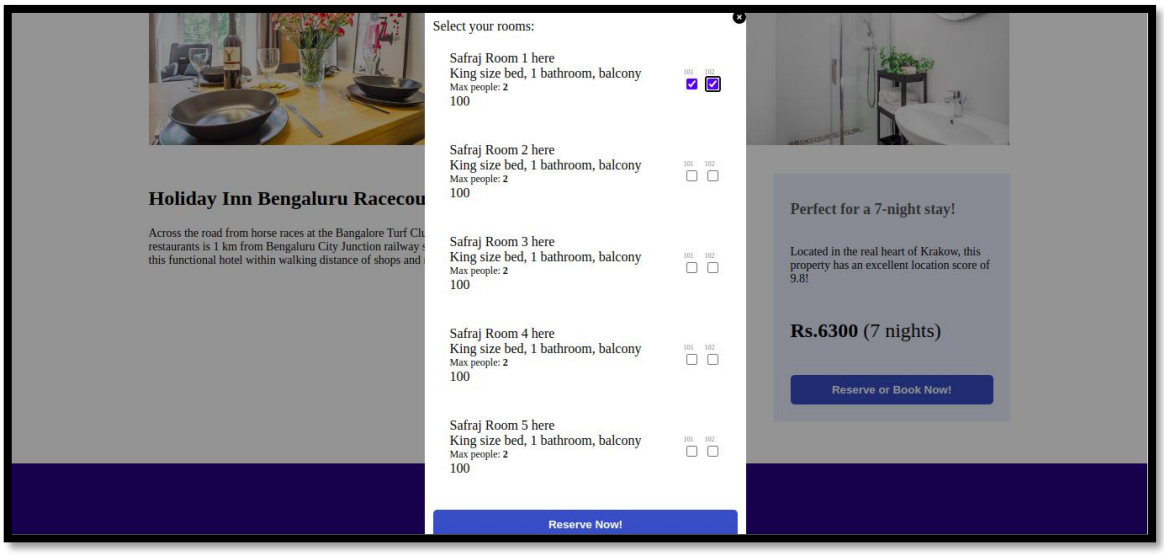
Future Improvements

- Implementing artificial intelligence (AI) and machine learning (ML) algorithms to provide users with personalized hotel recommendations based on their search history and preferences. This would improve the user experience by offering more relevant and customized results.
- Integration with virtual reality (VR) technology to provide users with immersive 3D tours of hotels, rooms, and amenities. This would enhance the user experience by giving users a more realistic sense of what the hotel is like before they book.
- Offering more flexible booking options, such as the ability to book specific room types or add-ons, or the option to book multiple rooms or split payment between multiple users. This would provide users with more control and flexibility over their bookings. Reminding of memories of an user through the email on the same day every year.
- Implementation of a loyalty program to reward users for their continued use of the platform. This would encourage users to book through the app more frequently and build a sense of brand loyalty.

Module Snippets







References

- https://www.w3schools.com/
- https://nodejs.org/en
- https://react.dev/