# Introduction

## Online Parking Slot Booking System

The proposed project is a smart parking booking system that provides customers an easy way of reserving a parking space online. It overcomes the problem of finding a parking space in commercial areas that unnecessary consumes time. Hence this project offers a web based reservation system where users can view various parking areas and select the space to view whether space is available or not. If the booking space is available then he can book it for specific time slot. The booked space will be marked yellow and will not be available for anyone else for the specified time. This system provides an additional feature of cancelling the bookings. User can cancel their books space anytime. Users can even make payment online via credit card. After making payment users are notified about the booking via email along with unique parking number.

# Modules:

- **Admin Login:** The system is under supervision of admin who manages the bookings made.

- **User login/registration:** Users have to first register themselves to login into the system.

- **Three Parking areas:** The system will provide users with three parking areas of different locations.

- **Parking availability check:** User can click on spaces to view the availability. If the space is already booked it will be marked yellow and the available ones will be seen in normal color.
- **Parking booking online for date and time:** Users can book parking space for their required date and time.

- **Automatic cost calculation:** The system calculates the total cost incurred for parking based on the time that user has asked for booking.

- **Parking cancellation:** User may even cancel their bookings by login into the system anytime.

- **Email on successful parking booking:** When user is successful in parking the space, system sends a confirmation and 'thank you' email regarding the space booked.

- **Feedback:** The system has a feedback form, where user can provide feedback into the system.

## User side functionality:

- Book parking space
- Cancellation
- Receipt Print
- Feedback
- Automatic calculate parking charge

# Admin side functionality:

- Administers parking booked
- Cancellation
- View User Data
- Feedback view and reply

## Software Requirements:

- React .js
- Node .js
- Visual studio code

### Language Uses In This Application

- HTML
- CSS
- BASIC + ADDVANCE JAVASCRIPT

## Hardware Components:

- Processor – i5
- Memory – 2GB RAM

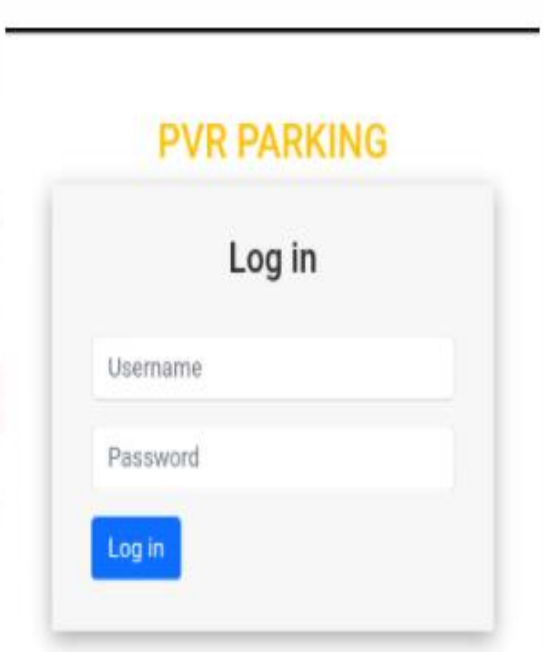
## Advantages:

- Users can get learn about parking areas for particular locations.
- It saves user time in search of parking space available in such a long parking area.
- The system provides a conceptual view of the parking spaces.
- User can pay online on the spot and confirm their space.
- It excludes the need of human efforts for managing parking spaces.
- The system generates online bill for requested time and even sends a text message through mobile number.
- Cost-effective.
- Paper less

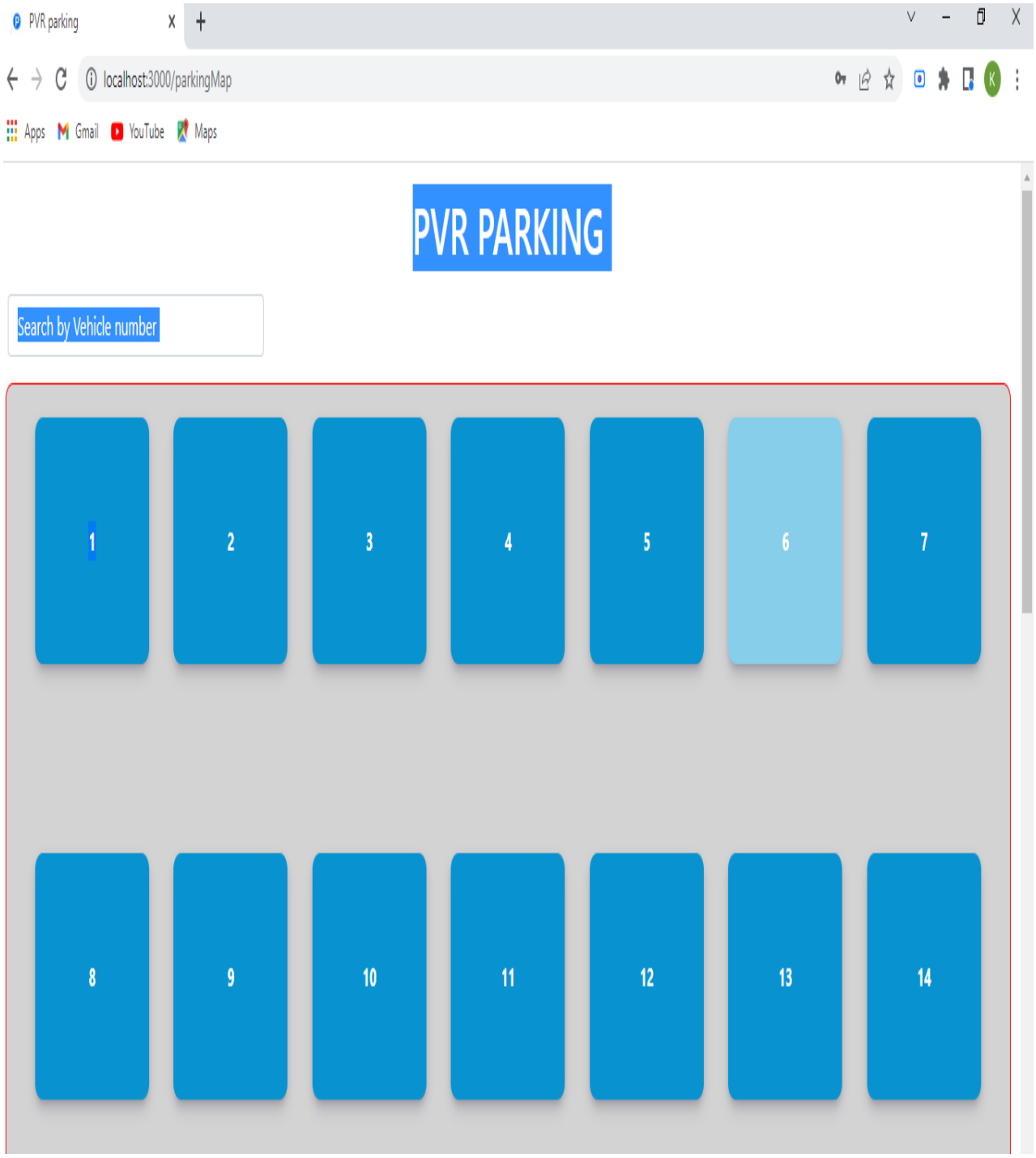
**Applications:**

- The project can be implemented in commercial areas for employee parking.
- It can be utilized by companies and organizations (**hospitals, schools, colleges**) to automate their parking system.
- The system can also be used in public places for public parking like in **malls, station,** and so on.

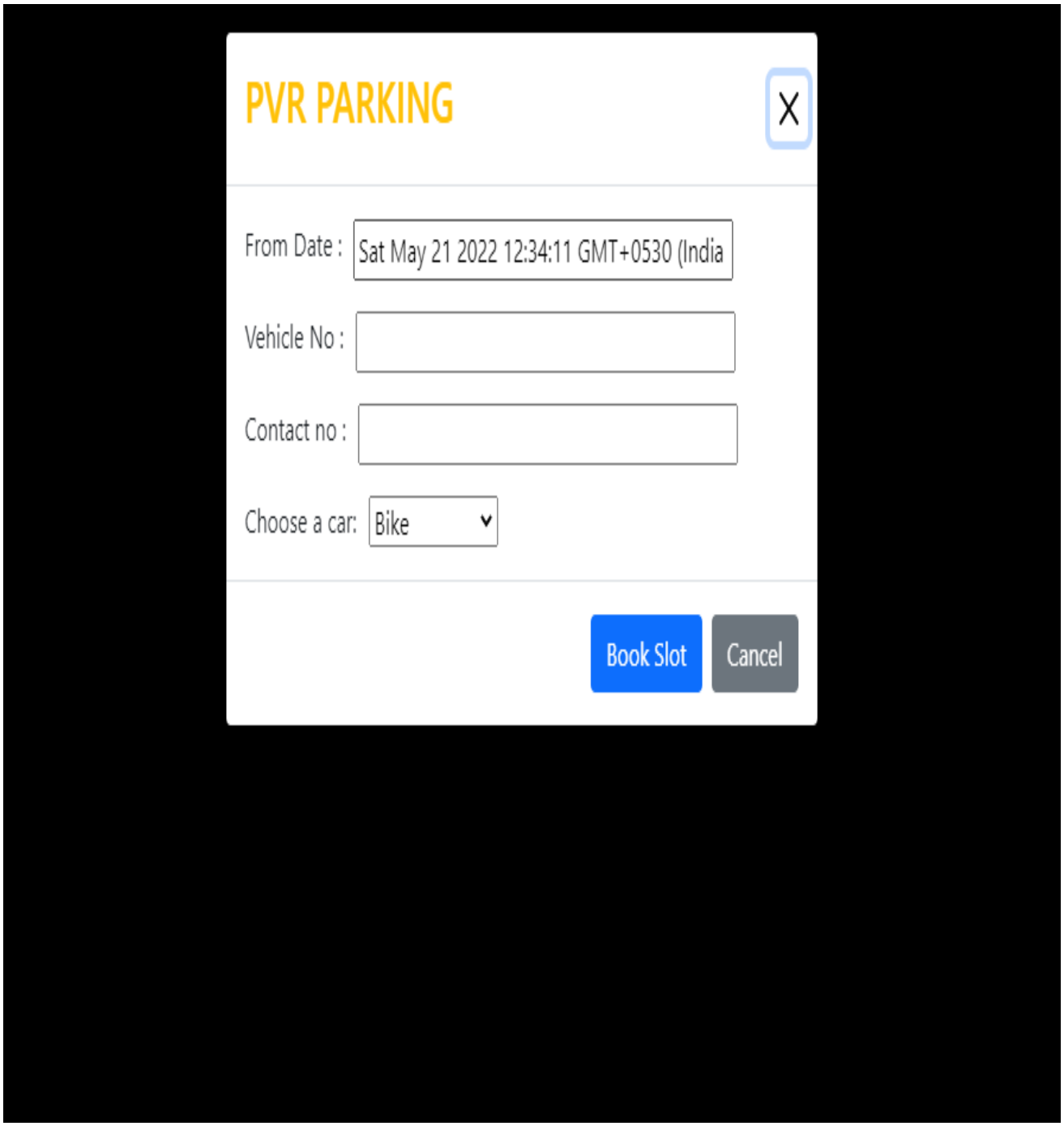
# Web Portal – Login And Home Screen



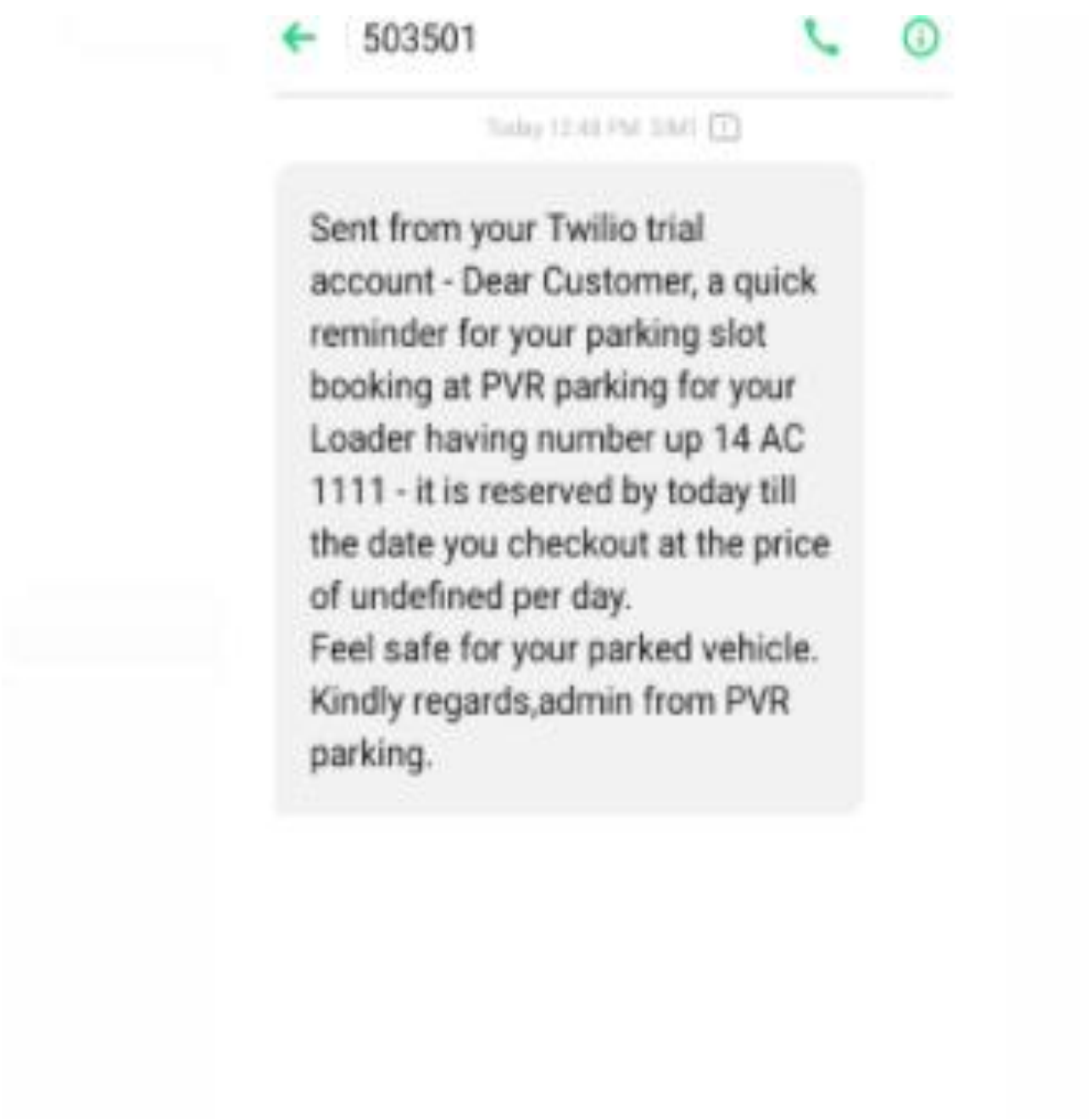
# Click On Login Button



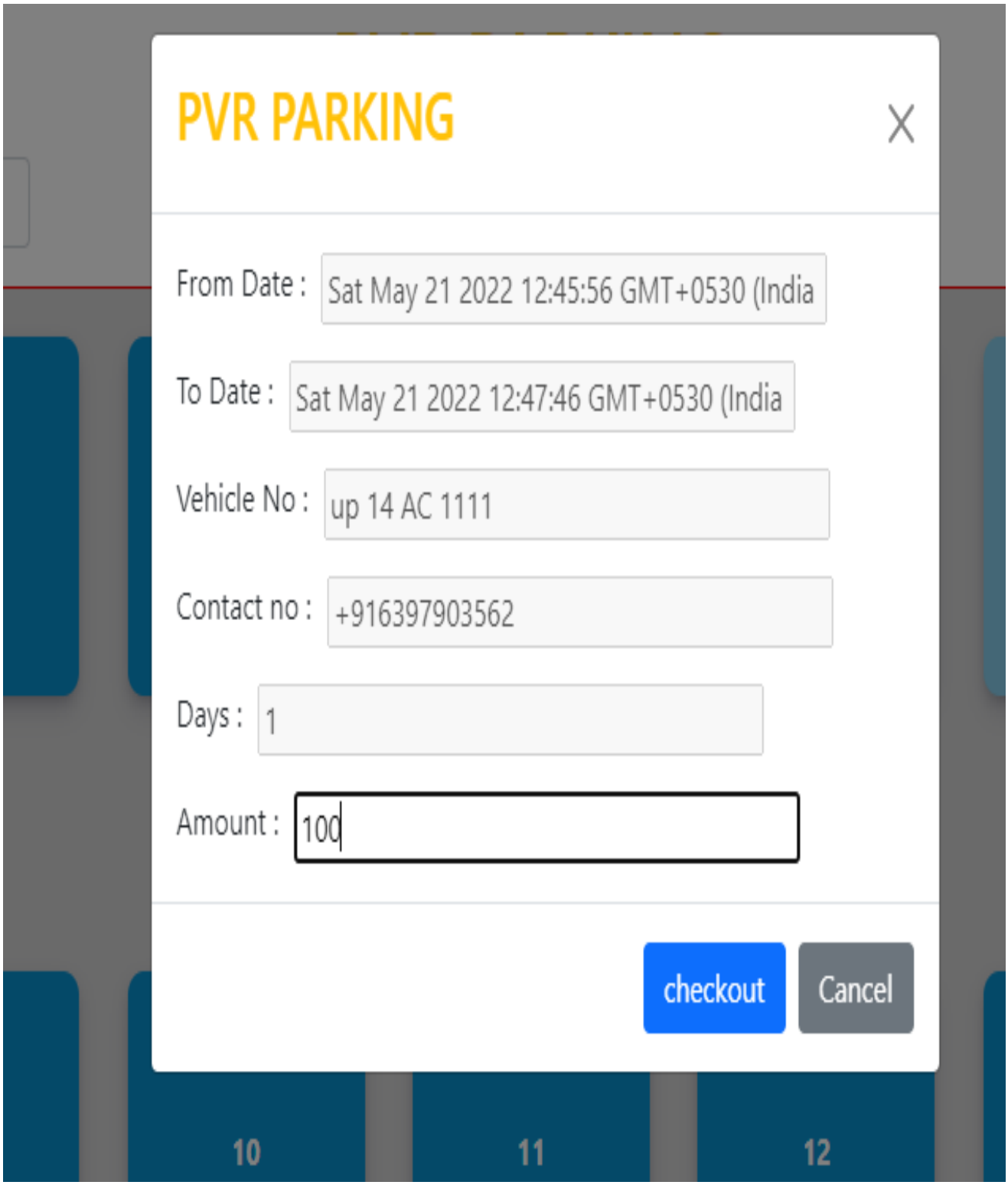
## Click on available slot:



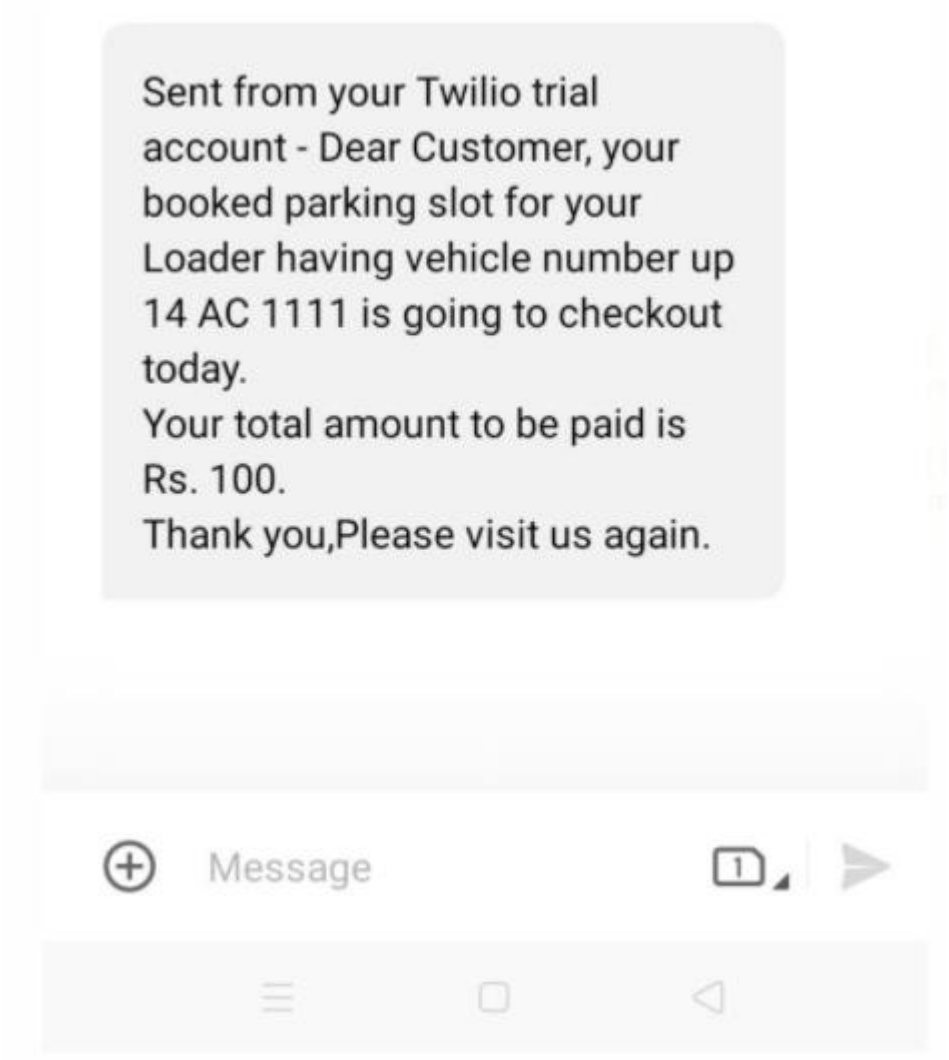
## Slot Booking Confirmation Message :



## Checkout Screen :



## Checkout confirmation message:




## Flow

# Chart Of Application:

Figure 1: below shows two flow chart that demonstrates how the system works.

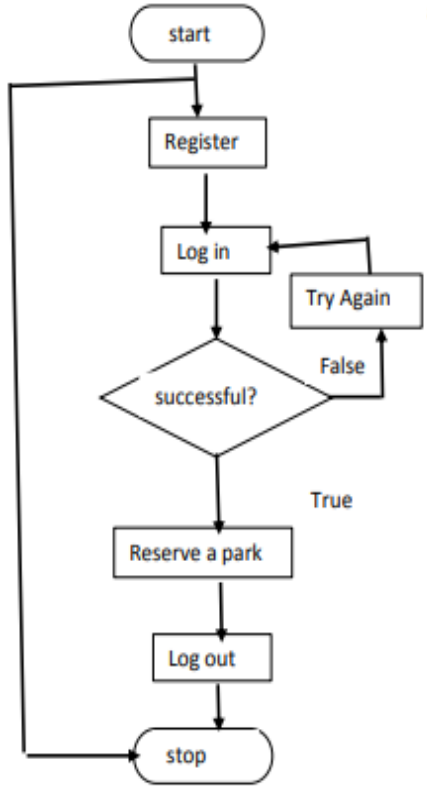
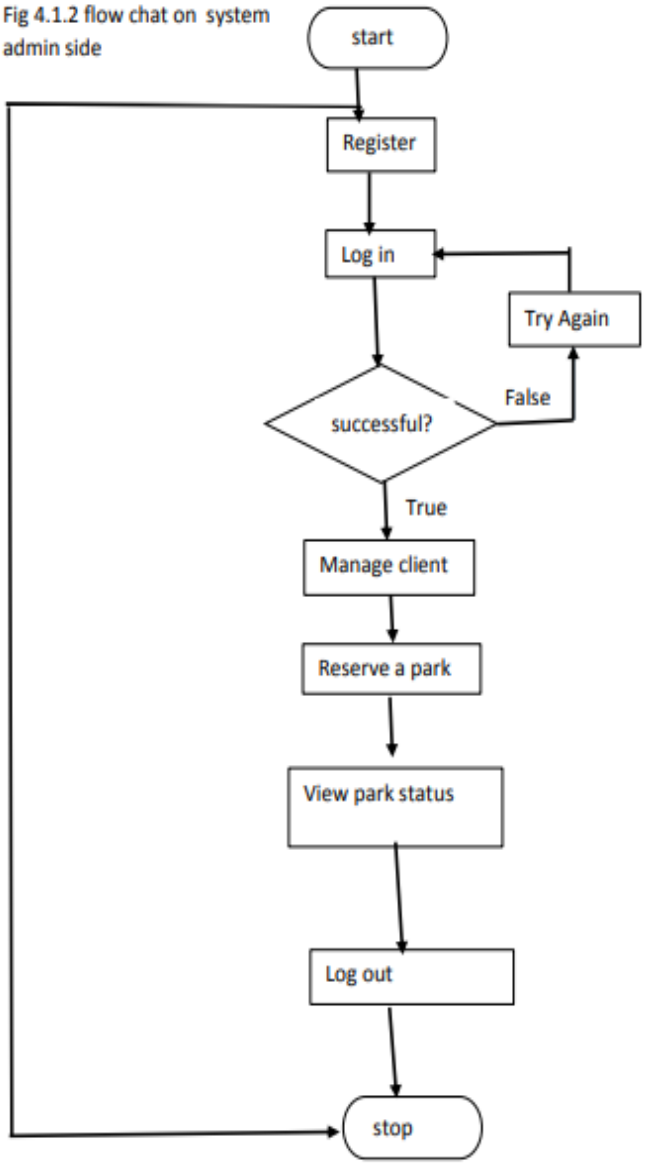



Fig 4.1.1:flow chart on client side

Fig 4.1.2 flow chat on system admin side



# Functional Requirements

- The web application displays the availability of parking lot
- The web application enables employees to set the reaching date and time for the car also the departure date and time.
- The web application enables employees to cancel a parking place.
- The web application enables drivers to book parking place with in PPK.

# Coding

## LoginPage.jsx

```
import React, { useState } from 'react'
import './login.css'
import { LoginAdmin } from
'../Service/postApi'
import { useNavigate } from 'react-router-
dom'
import { toast, ToastContainer } from 'react-
toastify';
import 'react-
toastify/dist/ReactToastify.css';

export default function Login() {

    const navi = useNavigate();

    const [adminLogin, setAdminLogin] =
useState(
        {
            username: "",
            password: ""
        }
    )

    const handleChange = (e) => {
        let key = e.target.name
        let val = e.target.value
```

```
        setAdminLogin({ ...adminLogin, [key]:
val })
    }
    const handleSubmit = async (e) => {
        e.preventDefault();
        const { username, password } =
adminLogin;
        if (username == "admin" && password
== "admin123") {

            toast('Login Successfull');
            sessionStorage.setItem('token',
'true');
            setTimeout(() => {
                navi('/parkingMap');
            }, 1000);

        } else {
            toast('Please Input Valid
Username/Password');

        }
    }
    return (
        <>
            <ToastContainer
                position="top-center"
                autoClose={2000}
                hideProgressBar={false}
                newestOnTop={false}
                closeOnClick
            />
            <div class="login-form">
```

```
                <h1 style={{ textAlign:
'center' }} className='text-warning'>PVR
PARKING </h1>
                <form className='d-flex flex-
column gap-3' onSubmit={handleSubmit}>
                    <h2 class="text-
center">Log in</h2>
                    <div class="form-
group  mt-10  ">
                        <input
                            name='username'
                            value={adminLogin
.username}
                            onChange={handleC
hange}
                            type="text"
class="form-control" placeholder="Username"
required="required" />
                    </div>
                    <div class="form-group
mt-10 ">
                        <input
                            name='password'
                            value={adminLogin
.password}
                            onChange={handleC
hange}
                            type="password"
class="form-control" placeholder="Password"
required="required" />
                    </div>
```

```
                    <div class="form-group
mt-10 ">
                        <button type="submit"
class="btn btn-primary btn-block">Log
in</button>
                    </div>
                </form>

            </div>


        </>)
}
```

## ParkedModel.jsx

```
   import React, {  useState } from 'react'

import { Button, Modal, ModalBody,
ModalFooter, ModalHeader } from 'reactstrap'
import { CheckOut } from
'../Service/updateApi'
export default function ParkedModal(props) {
     const
{visible,setVisible,setOpen,open,to_date,setS
lots} = props
     const
{from_date,vehicle_number,mobile_number,vehic
le_category,slot_number} = props.slot
```

```
      var date1 = new Date(from_date);
      var date2 = new Date(to_date);
      var Difference_In_Time = date2.getTime()
- date1.getTime();
      var Difference_In_Days =
Difference_In_Time / (1000 * 3600 * 24);
      Difference_In_Days=Math.ceil(Difference_
In_Days)
      const [amount, setAmount] = useState('')


const handleClick=()=>{
      setVisible(!visible);
      setOpen(!open);
      const data = {
        from_date,to_date,vehicle_category,amo
unt,vehicle_number,slot_number,mobile_number
      }
      console.log(data);
       CheckOut(data).then(res=>setSlots(res.d
ata.data)).catch(e=>console.log(e))
}

  return (
    <>
    <Modal
                  isOpen={visible}
                  toggle={() =>{
setVisible(!visible);setOpen(!open)}}
                >
                  <ModalHeader toggle={() =>{
setVisible(!visible);setOpen(!open)}}>
```

```
                    <h3 className='text-
warning'>PVR PARKING</h3>
                  </ModalHeader>
                  <ModalBody>
                         <div style={{ width:
'100%', display: 'flex', flexDirection:
'column', gap: '15px' }}>
                <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

                >
                  From Date : <input
type="text" style={{ width: '20rem' }}
                   disabled
                  name='from_date'
                  defaultValue={from_date}
                 />
                </div>
                <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

                >
                  To Date : <input type="text"
style={{ width: '20rem' }}
                  disabled
                  name='to_date'
                  defaultValue={to_date}
                 />
                </div>
                <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

                >
```

```
                Vehicle No : <input
type="text" style={{ width: '20rem' }}
                disabled
                 name='vehicle_number'
                 defaultValue={vehicle_number
}
            />
              </div>
              <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

              >
                Contact no : <input
type="text" style={{ width: '20rem' }}
                  disabled
                  name='mobile_number'
                  defaultValue={mobile_number
}
                />
              </div>
              <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

              >
                Days : <input type="text"
style={{ width: '20rem' }}
                  disabled
                  name='days'
                  Value={Difference_In_Days}
                />
              </div>
              <div style={{ width: '100%',
display: 'flex', gap: '10px' }}
```

```
                >
                    Amount : <input type="text"
style={{ width: '20rem' }}
                  onChange={(e)=>{setAmount(e.t
arget.value)}}
                    name='amount'
                    Value={amount}
                  />
                </div>
              </div>
                  </ModalBody>
                  <ModalFooter>
                    <Button
                      color="primary"
                      onClick={handleClick}

                    >
                    checkout
                    </Button>

                    <Button onClick={() =>{
setVisible(!visible);setOpen(!open)}}>
                      Cancel
                    </Button>
                  </ModalFooter>
                </Modal>
                </>
  )
}
```

# Parkingmap.jsx

```
import React, { useEffect, useState } from
'react'
import { getParkingSlots } from
'../Service/getApi'
import './parkingMap.css'
import ParkedModal from './parkedModal'
import UnParkedModal from './unparkedModal'
import HeavyTruck from '../images/delivery-
truck-front.png'
import Truck from '../images/truck.png'
import Car from '../images/sedan-car-
front.png'
import Bike from '../images/motorcycle.png'
import MiniTruck from
'../images/minitruck.png'
import Search from './search'


export default function Parkingmap() {

  const [slots, setSlots] = useState([])
```

```
  const [slotstatus, setslotStatus] =
useState('')
  const [visible, setVisible] =
useState(null)
  const [slot_number, setSlot_number] =
useState('')
  const [slotbooktime, setSlotbooktime] =
useState('')
  const [searchTerm, setSearchTerm] =
useState("")

  const [open, setOpen] = useState(false)

  useEffect(() => {

    getParkingSlots().then(res =>
setSlots(res.data.data)).catch(e =>
console.log(e))

  }, [])
  const handleSearch = e => {
    setSearchTerm(e.target.value)
    console.log(searchTerm);

}


  const handleClick = (slot, s_status, id) =>
{

    setVisible(!visible)
    setslotStatus(s_status)
    setOpen(!open)
```

```
    var BookDate = new Date();
    setSlotbooktime(BookDate);
    setSlot_number(slot);
  }
  return (
    <>
      <h1 className='m-auto text-center text-
warning mt-3 mb-3'>PVR PARKING</h1>
      <Search handleSearch={handleSearch}  />
      <div style={{
        backgroundColor: 'lightgray',
borderRadius: '10px',marginBottom:'5px',
        height: 'auto', width: '98%',
display: 'flex', gap: '15px', padding:
'20px', flexWrap: 'wrap', justifyContent:
'space-evenly', border: '1px solid red',
margin: 'auto'

      }}>

        {slots.filter((slot) => {
                        if (searchTerm ===
"")
                            return slot

                        else if
(slot.vehicle_number.toLowerCase().includes(s
earchTerm.toLowerCase()))
                            return slot

                    }).map((slot,id) => {
          return <>
            <div className='slots'
```

```
                style={{display:"flex",justifyC
ontent:'center',alignItems:'center',objectFit
:'contain',position:'relative',flexDirection:
'column',
                  borderRadius: '10px',
marginBottom: '100px', backgroundColor:
slot.slot_status ?
'skyblue':'rgb(8,146,208)',
                  height: '150px', minWidth:
'150px',color:'white'
                }}
                onClick={() =>
handleClick(slot.slot_number,
slot.slot_status, id)}>

              {slot.slot_status && <div
className='slot-hover'    >
                {slot.from_date}
                <br/>
                Slot No:{slot.slot_number}
              </div>}
              {slot.slot_status &&
slot.vehicle_category == "Car" && <img
src={Car} style={{ height: '50%', width:
'50%' }} />}
              {slot.slot_status &&
slot.vehicle_category == "Bike" && <img
src={Bike} style={{ height: '50%', width:
'50%' }} />}
              {slot.slot_status &&
slot.vehicle_category == "Loader" && <img
src={Truck} style={{ height: '50%', width:
'50%' }} />}
```

```
                {slot.slot_status &&
slot.vehicle_category == "Heavy Truck" &&
<img src={HeavyTruck} style={{ height: '50%',
width: '50%' }} />}
                {slot.slot_status &&
slot.vehicle_category == "Mini Truck" && <img
src={MiniTruck} style={{ height: '50%',
width: '50%' }} />}
              <b>{slot.slot_number}</b>
              </div>

              {slotstatus && open &&
(parseInt(slot_number) === parseInt(id)+1) &&
<ParkedModal visible={visible}
setVisible={setVisible}
                slot={slot}
                setSlots={setSlots}
                slot_number={slot_number}
                to_date={slotbooktime}
                setOpen={setOpen}
                open={open} />}

              {!slotstatus && open &&
<UnParkedModal visible={visible}
slot_num={slot_number} setSlots={setSlots}
                setOpen={setOpen}
                id={id}
                open={open}
                setVisible={setVisible}
from_date={slotbooktime} />}
            </>
          })}
        </div>
```

```
    </>
  )
}
```

# Searching.jsx

```
import React from 'react'


export default function Search(props) {

    return (
        <>
            <div className='d-flex justify-
content-end h-25 w-25 m-3 gap-2' style={{
placeItems: 'end' }}>

                <input type="text"
className='form-control'  placeholder='Search
by Vehicle
number'onChange={props.handleSearch}
/

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

# unparkedModel.jsx

```
import React, { useState } from 'react'
import { Button, Modal, ModalBody,
ModalFooter, ModalHeader } from 'reactstrap'
import { BookSlot } from
'../Service/updateApi'
import '../App.css'
export default function UnParkedModal(props)
{

console.log(props.visible);
  const { visible,
setVisible,from_date,slot_num,setOpen,open,se
tSlots} = props


  const [BookingDetails, setBookingDetails] =
useState({
    slot_number:`${slot_num}`,
    from_date: `${from_date}`,
    vehicle_number:'',
    mobile_number:'',
    vehicle_category:'Bike',

  })
```

```
  const handleClick = () => {
    setVisible(!visible)
    setOpen(!open)
    BookSlot(BookingDetails).then(res=>setSlo
ts(res.data.data))

  }

  const handleChange=(e)=>{
    let key=e.target.name;
    let val=e.target.value;
    setBookingDetails({...BookingDetails,[key
]:val})
  }
  console.log(BookingDetails);
  return (
    <>
      <div>

        <Modal isOpen={visible}
          toggle={() =>{
setVisible(!visible);setOpen(!open)}}>
            <ModalHeader toggle={() =>{
setVisible(!visible);setOpen(!open)}}>
                  <h3 className='text-
warning'>PVR PARKING</h3>
                </ModalHeader>

          <ModalBody style={{ height: "auto"
}}>
            <div style={{ width: '100%',
display: 'flex', flexDirection: 'column',
gap: '15px' }}>
```

```
                <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

                >
                  From Date : <input
type="text" style={{ width: '20rem' }}
                  name='from_date'
                  value={BookingDetails.from_da
te}
                  onChange={handleChange} />
                </div>
                <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

                >
                  Vehicle No : <input
type="text" style={{ width: '20rem' }}
                   name='vehicle_number'
                   value={BookingDetails.vehicl
e_number}
                  onChange={handleChange}/>
                </div>
                <div style={{ width: '100%',
display: 'flex', gap: '10px' }}

                >
                  Contact no : <input
type="text" style={{ width: '20rem' }}
                    name='mobile_number'
                    value={BookingDetails.mobil
e_number}
                  onChange={handleChange} />
                </div>
```

```
                <div style={{ width: '100%',
display: 'flex', gap: '10px' }}
                >
                 Choose a car:
                  <select
name="vehicle_category" id="vehicle_category"
onChange={handleChange}
value={BookingDetails.vehicle_category}>

                    <option value="Bike"
>Bike</option>
                    <option
value="Car">Car</option>
                    <option value="Heavy
Truck">Heavy Truck</option>
                    <option
value="Loader">Loader</option>
                    <option value="Mini
Truck">Mini Truck</option>
                  </select>
                </div>
              </div>
            </ModalBody>
            <ModalFooter>
              <Button
                color="primary"
                onClick={handleClick}
              >
                Book Slot
              </Button>

              <Button onClick={() =>{
setVisible(!visible);setOpen(!open)}}>
```

```
                Cancel
              </Button>
            </ModalFooter>
          </Modal>
        </div>
      </>
    )
}
```

**getApi.jsx**

```
import axios from "axios"
import { URL } from
"./endpoint"


 export  const getParkingSlots=
async ()=>{
    return  await
axios.get(`${URL}/api/PVR/parki
ng`)
}
```

**postApi.jsx**

```
import axios from "axios"
import { URL } from "./endpoint"


 export  const LoginAdmin= async
(adminDetail)=>{
    return await
axios.post(`${URL}/api/PVR/parking/login`,adm
inDetail)
}
```

# Update.jsx

```
import axios from "axios"
import { URL } from "./endpoint"


 export  const BookSlot= async
(bookingDetails)=>{
    return
await  axios.patch(`${URL}/api/PVR/parking/bo
oking`,bookingDetails)
}

export  const CheckOut= async (data)=>{
    return
await  axios.patch(`${URL}/api/PVR/parking/ch
eckout`,data)
}
```

# Protected.jsx

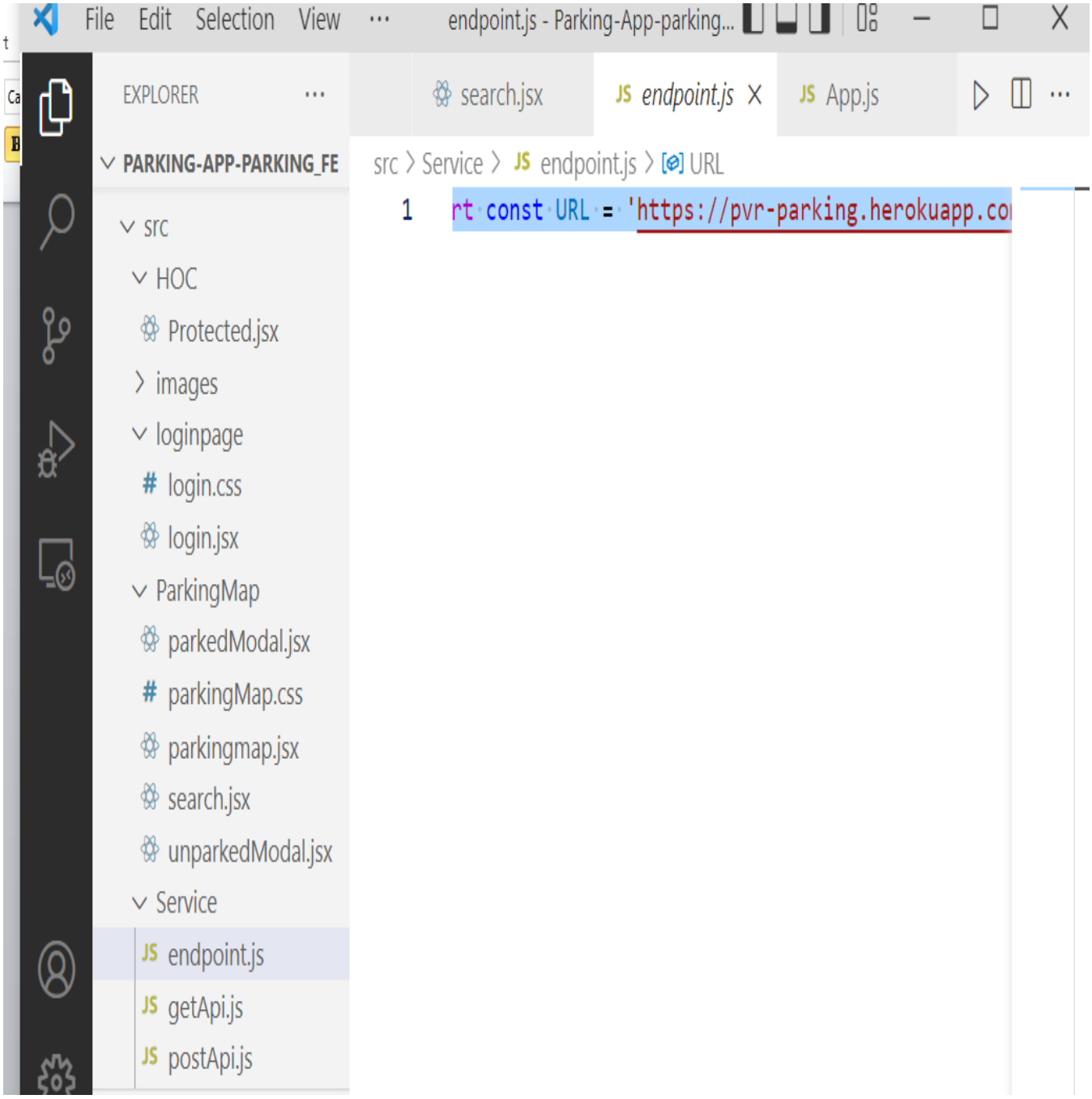
```
import React from 'react';
import {useNavigate} from 'react-router-dom'
import { toast } from 'react-toastify';

export function Protected({compo}) {

    const navigation = useNavigate();
       const [component, setComponent] =
              React.useState("");
    const token =
sessionStorage.getItem('token');

    React.useEffect(() => {
     token ? setComponent(compo) :
onUnauthorised();
    }, [])

    const onUnauthorised = () => {
      toast('Please Login First');
       navigation('/')
     }


    return component

}
```

# App.jsx

```
import { useRoutes } from 'react-router-dom'
import Login from './loginpage/login'
import Parkingmap from
'./ParkingMap/parkingmap'
import { Protected } from './HOC/Protected'
function App() {

  const routes = useRoutes ([
   {
     path:'/',element:<Login/>
   },
   {
    path:'/parkingMap',element: <Protected
compo={<Parkingmap/>}/>
  }
  ])
  return (
    <>
    {routes}

    </>
  )
}

export default App;
```

# Endpoint.js

```
export const URL = 'https://pvr-
parking.herokuapp.com'
```


File Edit Selection View ···
endpoint.js - Parking-App-parking...
EXPLORER
search.jsx
endpoint.js
App.js
PARKING-APP-PARKING_FE
src > Service > endpoint.js > URL
1 rt const URL = 'https://pvr-parking.herokuapp.co
src
HOC
Protected.jsx
images
loginpage
login.css
login.jsx
ParkingMap
parkedModal.jsx
parkingMap.css
parkingmap.jsx
search.jsx
unparkedModal.jsx
Service
endpoint.js
getApi.js
postApi.js

# TwilioSMS.js

```
const accountSid =
'AC81a771df88dafea49f95158000290800';
const authToken = 'S........';
const client = require('twilio')(accountSid,
authToken);


exports.sendMobileSMS = async ( body , to) =>
{
    return await client.messages.create({
        body,
        to,
        from: '+19704995417'
     }).then((r)=>console.log(r)).catch((e)=>
{console.log(e)})

}
```

# patchController.js

```
const path = require('path');
const fs = require('fs');
const parkingSlot = path.join(__dirname,
'../Storage/parkingSlot.json');
const { sendMobileSMS } =
require('../Utils/TwilioSMS')

exports.updateSlotStatus = async (req, res,
next) => {
    console.log(req.body);
    req.body.mobile_number = "+91" +
req.body.mobile_number
    const msg = `Dear Customer, a quick
reminder for your parking slot booking at PVR
parking - it is reserved by today till the
date you checkout.
Feel safe for your parked vehicle.
Kindly regards,admin from PVR parking.`
    const { slot_number,
from_date,  vehicle_number,  vehicle_category
, mobile_number } = req.body;
    const sentSMS = await sendMobileSMS(msg,
mobile_number)
    const data = fs.readFileSync(parkingSlot,
'utf8')
    const dataJson = data ? JSON.parse(data)
: [];
```

```
    const index = dataJson.findIndex((v) =>
v.slot_number == slot_number)
    if (index < 0) return next(new Error("No
Slot Exist"))
    dataJson[index].from_date = from_date
    dataJson[index].vehicle_number =
vehicle_number
    dataJson[index].vehicle_category =
vehicle_category
    dataJson[index].mobile_number =
mobile_number
    dataJson[index].slot_status = true
    fs.writeFile(parkingSlot,
JSON.stringify(dataJson), () => { })
    fs.readFile(parkingSlot, async (err,
dataSlot) => {
        if (err) return next(new
Error("Something went wrong"))
        res.status(201).send({ msg: 'slot
updated successfully', data:
JSON.parse(dataSlot) })
    })

}

exports.checkout = async (req, res, next) =>
{
    const { slot_number, from_date, to_date,
amount, vehicle_number, slot_status,
vehicle_category, mobile_number } = req.body;
    const msg = `Dear Customer, your booked
parking slot for your ${vehicle_category}
```

```
having vehicle number ${vehicle_number} is
going to checkout today.
Your total amount to be paid is Rs.
${amount}.
Thank you,Please visit us again.`
    const sentSMS = await sendMobileSMS(msg,
mobile_number)
    const data = fs.readFileSync(parkingSlot,
'utf8')
    const dataJson = data ? JSON.parse(data)
: [];
    const index = dataJson.findIndex((v) =>
v.slot_number == slot_number)
    if (index < 0) return next(new Error("No
Slot Exist"))
    dataJson[index].from_date = "",
    dataJson[index].to_date = "",
    dataJson[index].amount = "",
    dataJson[index].vehicle_number = "",
    dataJson[index].vehicle_category = "",
    dataJson[index].slot_status = false,
    dataJson[index].mobile_number = "";
    fs.writeFile(parkingSlot,
JSON.stringify(dataJson), () => { })
    fs.readFile(parkingSlot, async (err,
dataSlot) => {
        if (err) return next(new
Error("Something went wrong"))
        res.status(201).send({ msg: 'slot
updated successfully', data:
JSON.parse(dataSlot) })
    })}
```

# Main.js

```
const server =require('./server')

server.listen(process.env.PORT || 8080
,async()=>{
    console.log("Server is Started");
})
```