**22SDIN03R – FULLSTACK FOR IOT**

**Project**

**on**

**AUTOMOBILE ENUMERATION**

**by**

**2200100040 B.Dhinesh Reddy**



**KONERU LAKSHMAIAH EDUCATION FOUNDATION DEPARTMENT OF INTERNET OF THINGS**

Green Fields, Vaddeswaram, Guntur District-522 502

April- 2024

1

# ABSTRACT

* "Automobile Enumeration" is a web-based application developed using React, a popular JavaScript library for building user interfaces. The application aims to address the challenge of efficiently managing parking spaces in multi-floor parking lots or structures.
* "Automobile Enumeration" is a React application designed to manage parking slots across multiple floors. The application offers a user-friendly interface to visualize and interact with parking spaces, allowing users to monitor availability, park and vacate cars, and navigate between different floors.
* Implemented with React's state management, the application dynamically updates the status of parking slots based on user interactions. Users can easily identify available and occupied slots, facilitating efficient parking management.
* The application's interface comprises intuitive components such as buttons for floor selection, input fields for inputting parking statistics, and visually represented parking slots. These elements are styled using custom CSS, providing a cohesive and visually appealing user experience.
* Through its functionalities and visual presentation, "Automobile Enumeration" serves as a practical tool for managing parking spaces, enhancing efficiency, and improving the overall parking experience.
* Overall, "Automobile Enumeration" offers a comprehensive solution for parking space management, combining real-time data updates, intuitive user interface design, and responsive technology to streamline the parking experience in multi-floor parking facilities.

Top of Form

# INTRODUCTION

* In the realm of transportation engineering and urban planning, understanding the dynamics of vehicular movement is paramount. This understanding begins with the fundamental concept of vehicle count – a quantitative assessment of the number of vehicles traversing a particular point or area within a specified timeframe.
* Vehicle count data serves as the bedrock upon which transportation systems are analyzed, planned, and optimized. Whether it's a bustling city intersection, a suburban highway, or a rural road network, the ability to accurately quantify vehicular flow is indispensable for various purposes, including traffic management, infrastructure planning, and public policy formulation.
* This introduction delves into the significance of vehicle count data, shedding light on its role in addressing contemporary challenges such as traffic congestion, air pollution, and sustainable mobility. It explores the diverse methodologies employed to conduct vehicle counts, ranging from traditional manual tallies to sophisticated automatic traffic counters and advanced video analytics.
* Furthermore, the introduction underscores the multifaceted applications of vehicle count data in informing transportation policies, optimizing road networks, enhancing safety, and fostering sustainable urban development. By providing a comprehensive overview of vehicle count fundamentals, this introduction sets the stage for a deeper exploration of its methodologies, applications, and implications in the field of transportation engineering and urban planning.

# FRONTEND

**APP.JS**

import React, { useState } from 'react';

import './App.css'; // Assuming you have a CSS file for styling

function App() {

  const [parkingFloors, setParkingFloors] = useState([

    { floorNumber: 1, leftSlots: Array(5).fill(false), rightSlots: Array(5).fill(false) },

    { floorNumber: 2, leftSlots: Array(5).fill(false), rightSlots: Array(5).fill(false) },

  ]);

  const [selectedFloor, setSelectedFloor] = useState(null);

  const parkCar = (floorIndex, side, slotIndex) => {

    const updatedParkingFloors = [...parkingFloors];

    updatedParkingFloors[floorIndex][side][slotIndex] = true;

    setParkingFloors(updatedParkingFloors);

  };

  const leaveParking = (floorIndex, side, slotIndex) => {

    const updatedParkingFloors = [...parkingFloors];

    updatedParkingFloors[floorIndex][side][slotIndex] = false;

    setParkingFloors(updatedParkingFloors);

  };

  const handleFloorsClick = () => {

    setSelectedFloor(selectedFloor === null ? 1 : null);

  };

  const handleBackClick = () => {

    setSelectedFloor(null);

  };

  const handleNextPageClick = () => {

    setSelectedFloor(selectedFloor === 1 ? 2 : null);

  };

  return (

    <div className="App">

      {selectedFloor === null ? (

        <div className="container">

          <div className="title">Automobile Enumeration</div>

          <div className="block">S Block</div>

          <div className="name-box">Total No Slots <input type="NUMBER" placeholder="" /></div>

          <div className="name-box">occupied Slots <input type="NUMBER" placeholder="" /></div>

          <div className="name-box">Available Slots <input type="NUMBER" placeholder="" /></div>

          <div className="floor-container">

            <button className="floors-button" onClick={handleFloorsClick}>Floors</button>

          </div>

        </div>

      ) : (

        <div className='floor'>

        <div className="App">

          <button className="back-button" onClick={handleBackClick}>← Back</button>

          <h1>Floor {selectedFloor}</h1>

          <div className="parking">

            {/\* Left side parking slots \*/}

            <div className="left-side">

              <div className="slot-container">

                {parkingFloors[selectedFloor - 1].leftSlots.map((isOccupied, index) => (

                  <div

                    key={index}

                    className={isOccupied ? 'slot occupied' : 'slot vacant'}

                    onClick={() =>

                      isOccupied

                        ? leaveParking(selectedFloor - 1, 'leftSlots', index)

                        : parkCar(selectedFloor - 1, 'leftSlots', index)

                    }

                  >

                    Floor {selectedFloor} Slot {index + 1}

                  </div>

                ))}

              </div>

            </div>

            {/\* Road separator \*/}

            <div className="road"></div>

            {/\* Right side parking slots \*/}

            <div className="right-side">

              <div className="slot-container">

                {parkingFloors[selectedFloor - 1].rightSlots.map((isOccupied, index) => (

                  <div

                    key={index}

                    className={isOccupied ? 'slot occupied' : 'slot vacant'}

                    onClick={() =>

                      isOccupied

                        ? leaveParking(selectedFloor - 1, 'rightSlots', index)

                        : parkCar(selectedFloor - 1, 'rightSlots', index)

                    }

                  >

                    Floor {selectedFloor} Slot {index + 6}

                  </div>

                ))}

              </div>

            </div>

          </div>

          {selectedFloor === 1 && (

            <button className="next-button" onClick={handleNextPageClick}>

              Next Page →

            </button>

          )}

        </div>

        </div>

      )}

    </div>

  );

}

export default App;

**APP.CSS**

.App {

  text-align: center;

  background-color: rgba(64, 69, 156, 0.235);

  background-position: 100px;

  margin: 10px;

}

.parking {

  display: flex;

  justify-content: center;

  align-items: center;

  width: 100%;

}

.left-side,

.right-side {

  width: 20%;

  display: flex;

  justify-content: center;

  align-items: center;

}

.slot-container {

  column-width: 1000px;

  display: flex;

  flex-direction: column;

  transform: translate();

}

.slot {

  border-radius: 200px;

  width: 120px; /\* Width of the rectangle slots \*/

  height: 60px; /\* Height of the rectangle slots \*/

  border: 2px solid #000;

  margin: 15px;

  display: flex;

  justify-content: center;

  align-items: center;

  cursor: pointer;

}

.occupied {

  background-color: red;

  color: rgb(0, 0, 0);

}

.vacant {

  background-color: green;

  color: rgb(0, 0, 0);

}

.container {

  max-width: 350px;

  max-height:250px;

  margin: 0 auto;

  color: rgb(255, 255, 255);

  padding: 190px;

  background-color:transparent;

  border-radius: 10px;

  box-shadow: 0 0 10px rgba(205, 62, 62, 0.1);

  background-image:url('https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSpvRbBR2FZS8FBN580hhX-LfHoliGhEehJtA&s');

  background-size: cover;

  background-position: center;

}

.title {

  text-align: center;

  font-weight: bolder;

  font-style: oblique;

  font-size: 50px;

  margin: -115px;

  margin-bottom: 2px;

}

.subheading {

  text-align: center;

  font-size: 50px;

  margin: 2px;

  margin-top: 1px;

  position: absolute;

  top: 100px;

  left: 50%;

  transform: translateX(-50%);

}

.block {

  background-color: transparent;

  font-size: larger;

  font-weight: bolder;

  font-style: oblique;

  padding: 20px;

  margin-bottom: 20px;

  border-radius: 5px;

  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.name-box {

  background-color: transparent;

  font-size: larger;

  font-weight: bolder;

  color: #fff;

  padding: 20px;

  margin-bottom: 20px;

  align-items: center;

  justify-content: center;

  border-radius: 5px;

  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.floor-container {

  display: flex;

  flex-direction: column;

}

.floor-container .name-box {

  display: flex;

  align-items: center;

}

.floor-container .name-box button {

  padding: 10px 20px;

  background-color: #000000;

  color: #fff;

  border: none;

  border-radius: 5px;

  margin-right: 10px;

  cursor: pointer;

}

.floor-container .name-box button:hover {

  background-color: #000000;

}

.floors-button{

background-color: skyblue;

font-size: 20px;

cursor: pointer;

padding: 10px 20px;

font-weight: bolder;

}

.back-button {

  position: absolute;

  top: 10px;

  left: 10px;

  color: #fff;

  background-color: rgb(0, 0, 0);

  font-size: 16px;

  padding: 10px 20px;

  cursor: pointer;

}

.next-button {

  position: absolute;

  top: 10px;

  right: 20px;

  color: #fff;

  background-color: #000000;

  padding: 10px 20px;

  font-size: 16px;

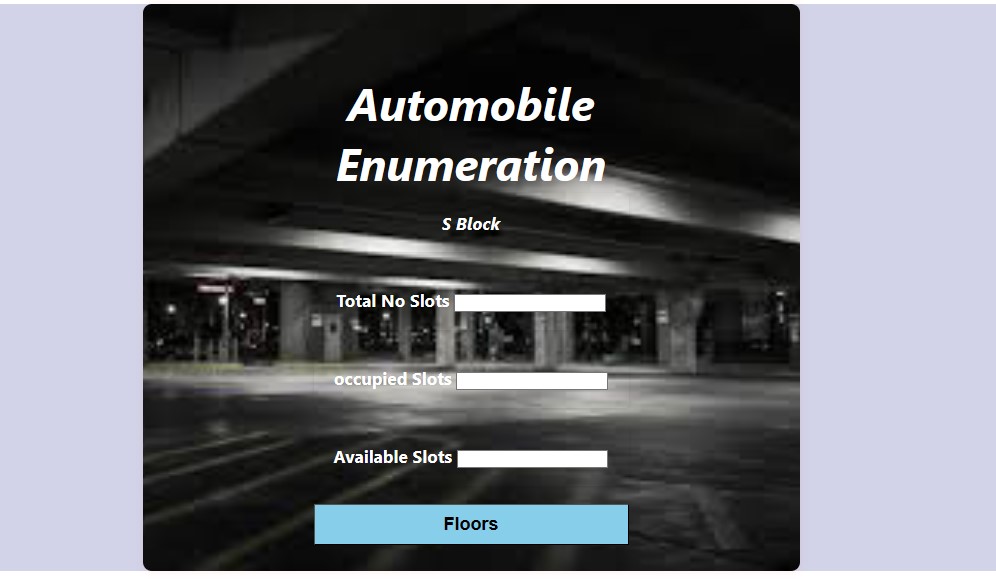
  cursor: pointer;

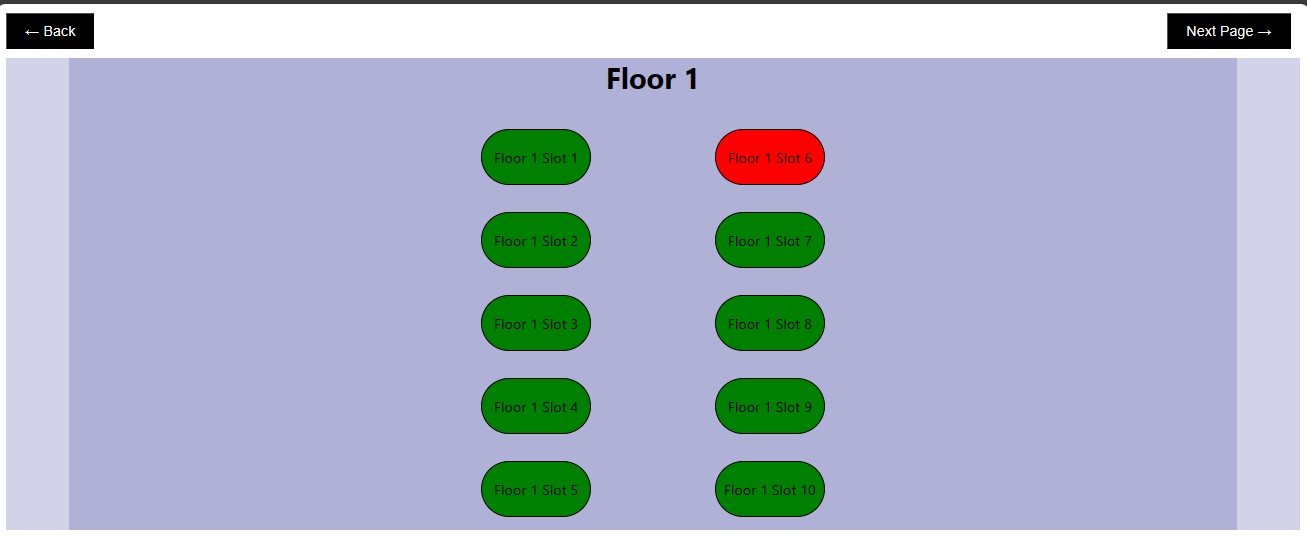
}

.floor{

  margin: 60px;

}





# BACKEND AND HARDWARE

#include <WiFi.h>

#include <WebServer.h>

const char\* ssid = "Rohitha";

const char\* password = "chaluikachalu";

const int irSensorPin1 = 32;

const int irSensorPin2 = 34;

WebServer server(80);

void setup() {

  Serial.begin(9600);

  pinMode(irSensorPin1, INPUT);

  pinMode(irSensorPin2, INPUT);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

    Serial.println("Connecting to WiFi...");

  }

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);

  server.begin();

  Serial.println("HTTP server started");

}

int sta1=0,sta2=0,count=0;

void loop() {

  server.handleClient();

  int se1 = digitalRead(irSensorPin1);

  int se2 = digitalRead(irSensorPin2);

  if(se1==1){

    sta1=1;

  }

  if(se1==0&sta1==1){

    count=count+1;

    sta1=0;

  }

  if(se2==1){

    sta2=1;

  }

  if(se2==0&sta2==1){

    count=count-1;

    sta2=0;

  }

  digitalWrite(se1,0);

  digitalWrite(se2,0);

  /\*

  if (count<0){

    count=0;

  }\*/

  delay(20);

}

void handleRoot() {

  int sensorState1 = digitalRead(irSensorPin1);

  int sensorState2 = digitalRead(irSensorPin2);

  String page = "<html><body>";

  page += "<h1>IR Sensor Status</h1>";

  page += "<p>vehical count ";

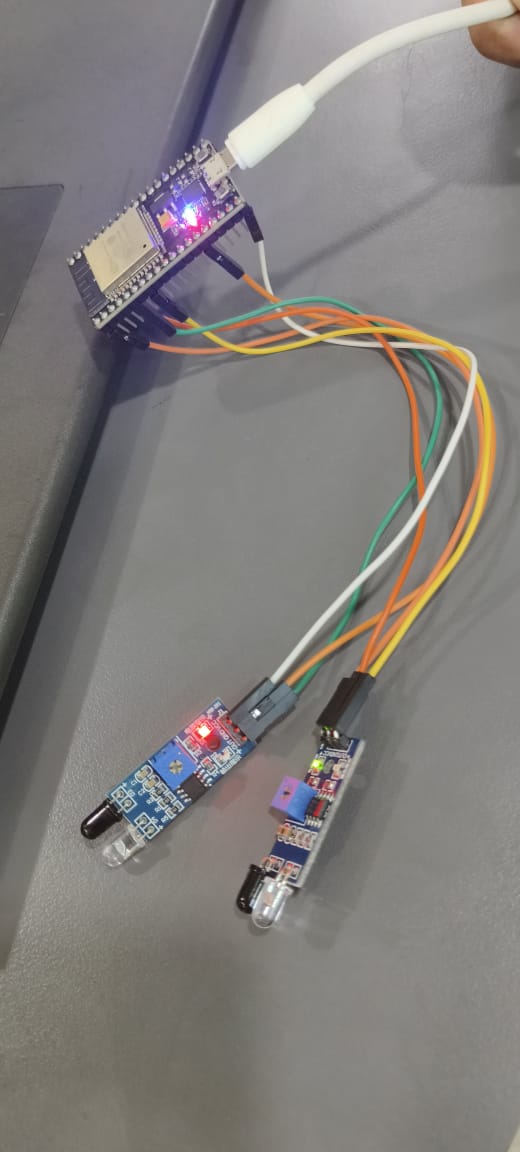
  page += count;

  page += "</p>";

  page += "</body></html>";

  server.send(200, "text/html", page);

}



**RESULT AND ANALYSIS**

# 

1. **Efficient Parking Management**: "Automobile Enumeration" provides an effective solution for managing parking spaces in multi-floor parking lots. Users can easily monitor the availability of parking slots, facilitating efficient utilization of parking resources.
2. **Real-time Updates**: The application's ability to update parking slot statuses in real-time ensures that users have access to the latest information. This feature enhances user experience and helps in making timely decisions while parking.
3. **User-Friendly Interface**: With its intuitive user interface, "Automobile Enumeration" offers a seamless experience for users. The layout is clear and easy to navigate, allowing users to quickly locate available parking slots and perform necessary actions.
4. **Responsive Design**: The application's responsive design enables it to adapt to various screen sizes and devices. Whether accessed on desktops, tablets, or smartphones, users can enjoy a consistent and optimized experience.
5. **Visual Representation**: The visual representation of parking slots using custom styling enhances user understanding. Users can easily differentiate between occupied and vacant slots, enabling them to make informed decisions.
6. **Customization Options**: The inclusion of customization options, such as input fields for parking statistics, adds flexibility to the application. Users can tailor the application to suit their specific needs or requirements.

Overall, "Automobile Enumeration" achieves its goal of simplifying parking space management in multi-floor parking facilities. Its combination of real-time updates, user-friendly interface, and responsive design makes it a valuable tool for optimizing parking resource utilization and enhancing the overall parking experience. Further analysis could involve user feedback and metrics on parking efficiency to assess the application's impact and identify areas for improvement.

Top of Form

# CONCLUSION

* "Automobile Enumeration" presents a comprehensive solution for managing parking spaces in multi-floor parking facilities. Through its real-time updates, intuitive interface, and responsive design, the application streamlines the parking experience for users, facilitating efficient utilization of parking resources.
* By providing users with accurate information about parking slot availability, the application helps reduce the time spent searching for parking spaces and minimizes congestion within parking structures. Its visual representation of parking slots and customization options contribute to a user-friendly experience, enhancing user satisfaction and usability.
* Overall, "Automobile Enumeration" demonstrates the potential of technology to improve everyday tasks such as parking management. As cities continue to grapple with urban congestion and limited parking availability, solutions like "Automobile Enumeration" can play a vital role in optimizing parking utilization and improving the overall urban mobility experience. Continued development and refinement of such applications are essential for addressing evolving parking challenges and enhancing urban livability.