SMART PARKING(IOT)

NAME: PRAVIN V

ID: AUT962921104303

EM AIL:vpravinpravin2001@gmail.com

PHASE:5

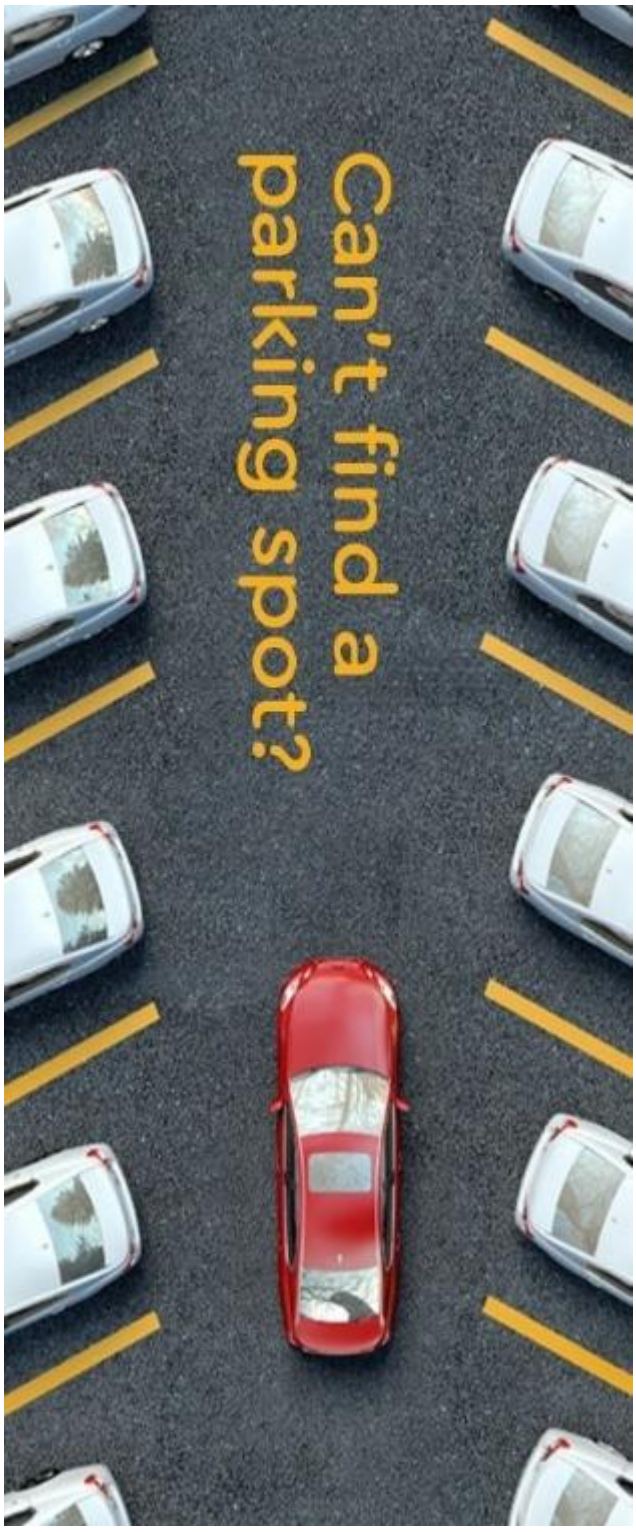


TABLE OF CONTENTS

INTRODUCTION:

Smart parking is a modern technological solution aimed at revolutionizing the way we approach parking in urban and suburban environments. As cities around the world face increasing challenges related to traffic congestion, limited parking spaces, and environmental concerns, smart parking systems have emerged as a promising answer to these issues. Smart parking leverages a combination of advanced technologies such as sensors, data analytics, mobile apps, and automation to provide a more efficient and user-friendly parking experience. The key idea behind smart parking is to optimize the utilization of parking spaces, reduce the time and fuel wasted in the search for parking, and enhance the overall urban mobility and quality of life. In a smart parking system, sensors are installed in parking spaces to monitor their availability in real-time. This data is then relayed to users through mobile applications or digital displays, allowing them to easily find and reserve parking spots.

Additionally, smart parking solutions often include features like automated payment systems and integration with navigation apps to guide drivers to available parking spaces. The benefits of smart parking extend beyond individual convenience. By reducing traffic congestion and the environmental impact of circling for parking, these systems contribute to improved air quality and reduced emissions. They also have the potential to boost revenue for municipalities or private operators through dynamic pricing and efficient enforcement. Furthermore, smart parking aligns with broader urban development goals by enhancing safety, promoting sustainable transportation alternatives, and integrating with other aspects of urban infrastructure, such as public transportation and city planning. In this age of increasing urbanization, where efficient use of space and resources is paramount, smart parking is a critical component of the smart city concept. It represents a proactive and innovative approach to tackling the parking challenges that accompany urban growth, making cities more livable and sustainable for residents and visitors alike. As technology continues to advance, the future of smart parking promises even more innovative and intelligent solutions to improve the way we park and move within urban environments.

PROJECT OBJECTIVES:

Optimize Parking Space Utilization:

The primary goal of smart parking is to maximize the use of available parking spaces. This involves reducing congestion and minimizing the time and fuel wasted by drivers searching for parking spots.

Reduce Traffic Congestion: By guiding drivers to available parking spaces and reducing the time spent circling for a spot, smart parking systems aim to alleviate traffic congestion in urban areas. This can lead to reduced emissions and improved air quality.

Enhance User Convenience: Smart parking systems should make it easier for drivers to find, reserve, and pay for parking. Mobile apps, online booking, and real-time availability updates contribute to a more convenient and user-friendly experience.

Improve Revenue Generation: Many smart parking initiatives are designed to increase revenue for municipalities or private operators. This can be achieved through dynamic pricing, efficient enforcement, and improved space turnover.

Enhance Safety and Security: Smart parking solutions can incorporate security features like surveillance cameras and emergency call buttons to enhance the safety of parking facilities.

Reduce Environmental Impact: Reducing the time vehicles spend idling and circling for parking spots can contribute to lower fuel consumption and emissions, thus helping to combat air pollution and climate change.

Promote Sustainable Transportation: Smart parking projects often aim to encourage the use of public transport, carpooling, and non-motorized modes of transportation by making these options more accessible and convenient.

Data Collection and Analysis: Gathering data on parking space utilization, traffic patterns, and user behavior is a key objective. Analyzing this data can help urban planners make informed decisions and optimize parking policies.

Enhance Accessibility: Smart parking should be designed to cater to the needs of all users, including those with disabilities, by providing accessible parking spaces and user-friendly features.

Integration with Urban Infrastructure: Smart parking systems should be integrated with broader urban infrastructure, including traffic management, public transportation, and city planning, to ensure a cohesive and well-coordinated approach to urban mobility.

Sustainability and Green Initiatives: Some smart parking projects may include the implementation of sustainable infrastructure elements like electric vehicle charging stations, solar-powered parking meters, and green urban design.

Economic Development: In certain cases, smart parking projects are expected to stimulate economic growth by making it easier for people to access businesses and attractions in urban areas.

Enforcement and Compliance: Ensure that parking rules and regulations are effectively enforced through automated methods, such as license plate recognition or ticketing systems.
Feedback and User Engagement: Smart parking projects should encourage user feedback and engagement to continuously improve the system based on user experiences and preferences.

Scalability and Adaptability: The system should be designed to adapt to changing urban needs and to be scalable as the city or area grows.

IoT Devices :

Hardware components:

1. ESP8266 NodeMCU
2. Ultrasonic Sensor
3. DC Servo Motor
4. IR Sensors

5. 16x2 i2c LCD Display
6. Raspberry Pi
7. Jumpers

SOFTWARE :

1. Raspberry Pi OS
2. Python
3. GPIO Library

SENSOR SETUP:

Setting up an IoT project using an ESP8266 NodeMCU board, ultrasonic sensor, DC servo motor, IR sensors, a 16x2 I2C LCD display, and jumpers involves several steps. I'll provide an overview of how you can set up this project, but please note that this is a complex project, and you may need to consult specific documentation and libraries for each component. Additionally, coding this project will require programming skills in platforms like Arduino IDE.

1. Gather the Required Components:

- ESP8266 NodeMCU board.
- Ultrasonic sensor (e.g., HC-SR04).
- DC servo motor.
- IR sensors (for object detection).
- 16x2 I2C LCD display.
- Jumper wires and breadboard.
- Power supply for the servo motor if needed.

2. Connect the Ultrasonic Sensor:

- Connect the VCC pin of the ultrasonic sensor to the 3.3V output of NodeMCU.
- Connect the GND pin of the ultrasonic sensor to the GND of NodeMCU.
- Connect the TRIG pin of the ultrasonic sensor to a GPIO pin (e.g., D2).
- Connect the ECHO pin of the ultrasonic sensor to another GPIO pin (e.g., D3).

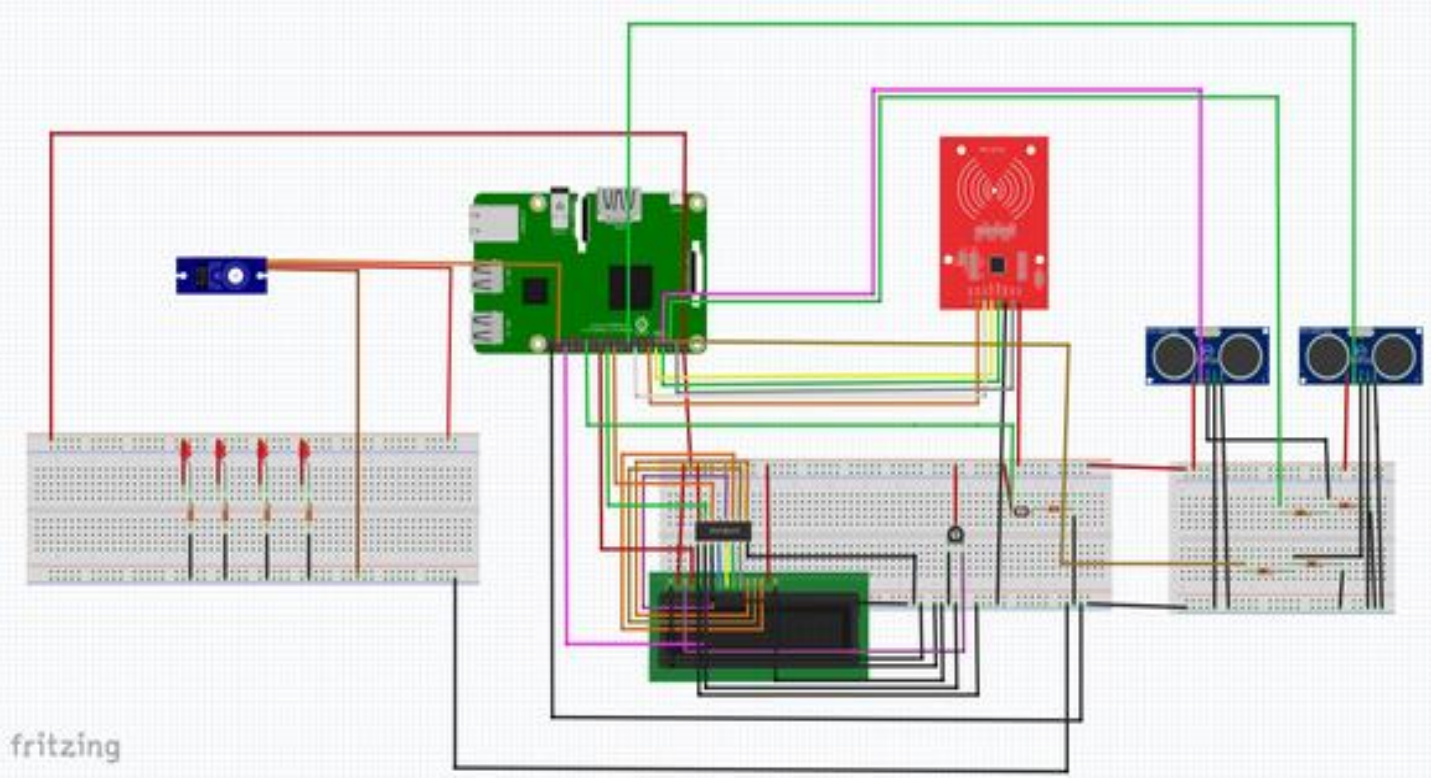
3. Connect the DC Servo Motor:
   - Connect the positive (red) lead of the servo motor to the 5V output of NodeMCU.
   - Connect the negative (brown) lead of the servo motor to the GND of NodeMCU.
   - Connect the signal (orange/yellow) lead of the servo motor to a GPIO pin (e.g., D4).
4. Connect the IR Sensors:
   - IR sensors are usually analog sensors. Connect the VCC and GND pins to 3.3V and GND on the NodeMCU.
   - Connect the signal pin of the IR sensors to analog GPIO pins (e.g., A0 and A1)
5. Connect the 16x2 I2C LCD Display:
   - Connect the SDA (data) and SCL (clock) pins of the I2C LCD display to the corresponding pins on the NodeMCU (D1 and D2 on the NodeMCU, respectively).
   - Connect the VCC of the I2C display to 5V on NodeMCU and GND to GND.
6. Write and Upload the Code:

   Write the Arduino code to control your project. This code will involve reading data from the ultrasonic sensor, processing it, controlling the servo motor, and displaying information on the LCD. You'll also need code to handle IR sensor inputs if they're used for object detection.
7. Power Supply:

   Make sure you have a suitable power supply for your servo motor, as the NodeMCU might not be able to provide enough power for it.
8. Assemble and Test:

Connect all the components, upload the code to the NodeMCU, and assemble theproject. Test each component and ensure that the system functions as expected.

MODEL:



MOBILE APP DEVELOPMENT:

1. Define Your Goals and Objectives:

• Clearly define the objectives and goals of your smart parking platform. Understand what problems you want to solve, whether it's reducing congestion, enhancing revenue generation, improving accessibility, or promoting sustainability.

2. Hardware Selection:

   • Choose the appropriate hardware components, such as sensors (ultrasonic, infrared, camera-based), microcontrollers (like Raspberry Pi or Arduino), communication modules (Wi-Fi, LoRa, or cellular), and displays for realtime information.

3. Sensor Installation:

   • Install sensors in parking spaces to monitor availability. Ensure the sensors can detect the presence or absence of vehicles accurately. These sensors may be ultrasonic or infrared-based, depending on your project requirements.

4. Communication Infrastructure:

   • Set up a reliable communication infrastructure that connects the sensors to a central server or cloud platform. Wi-Fi, LoRa, or cellular networks are commonly used for data transmission.

5. Central Server or Cloud Platform:

   • Develop a central server or cloud-based platform to collect, process, and store data from the sensors. Use a robust database system to manage realtime parking space availability.

6. User-Facing Mobile and Web Applications:

   • Create user-friendly mobile and web applications that allow users to:

   • Find available parking spaces.

   • Reserve parking spots in advance.

   • Make payments for parking.

   • Receive real-time updates on parking availability and guidance.

7. Payment Integration:

   • Integrate payment gateways into the mobile app for user convenience. This can include options for cashless payments, credit card payments, and mobile wallets.

8. Data Analytics and Prediction:

   • Implement data analytics to analyze historical and real-time parking data. This can help in predicting parking demand, optimizing pricing, and improving operational efficiency.

9. Security and Access Control:

• Ensure the security of data and transactions. Implement user authentication and access control features to prevent unauthorized use or tampering with the system.

10. Real-time Displays and Signage:

   • Install displays at the parking facility entrances to guide drivers to available spaces and provide real-time updates on space availability.

11. Environmental Considerations:

   • For sustainability, consider incorporating features such as electric vehicle charging stations, solar-powered components, and green infrastructure.

12. Scalability and Flexibility:

   • Design your platform to be scalable, allowing for easy expansion to more parking areas as needed.

13. Regulatory Compliance:

   • Ensure your platform complies with local parking regulations, privacy laws, and other relevant regulations.

14. Testing and Maintenance:

   • Thoroughly test the system to ensure its reliability and accuracy. Develop a maintenance plan for regular sensor and system checks.

15. User Education and Onboarding:

   • Provide clear instructions to users on how to use the platform and make the transition to smart parking smooth.

16. Marketing and Adoption:

- Promote your smart parking platform to attract users and encourage adoption. This may involve partnerships with local businesses and marketing campaigns.

17. Continuous Improvement:

- Continuously monitor the system's performance and gather user feedback for ongoing improvement and innovation.

CODE IMPLEMENTATION:

Program:

```
#include <ESP8266WiFi.h>
#include <Servo.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <FirebaseArduino.h>
#define FIREBASE_HOST "smart-parking-7f5b6.firebaseio.com" // the
project name address from firebase id
#define FIREBASE_AUTH
"suAkUQ4wXRPW7nA0zJQVsx3H2LmeBDPGmfTMBHCT" // the secret
key generated from firebase
#define WIFI_SSID "CircuitDigest" // input your home
or public wifi name
#define WIFI_PASSWORD "circuitdigest101" //password
for Wifi
String Available = ""; //availability string
String fireAvailable = "";
LiquidCrystal_I2C lcd(0x27, 16, 2); //i2c display address 27 and 16x2 lcd
display
Servo myservo; //servo as gate
Servo myservos; //servo as gate
int Empty; //available space integer
int allSpace = 90;
int countYes = 0;
int carEnter = D0; // entry sensor
int carExited = D4; //exi sensor
int TRIG = D7; //ultrasonic trig pin
int ECHO = D8; // ultrasonic echo pin
int led = D3; // spot occupancy signal
int pos;
int pos1;
long duration, distance;
void setup() {
```

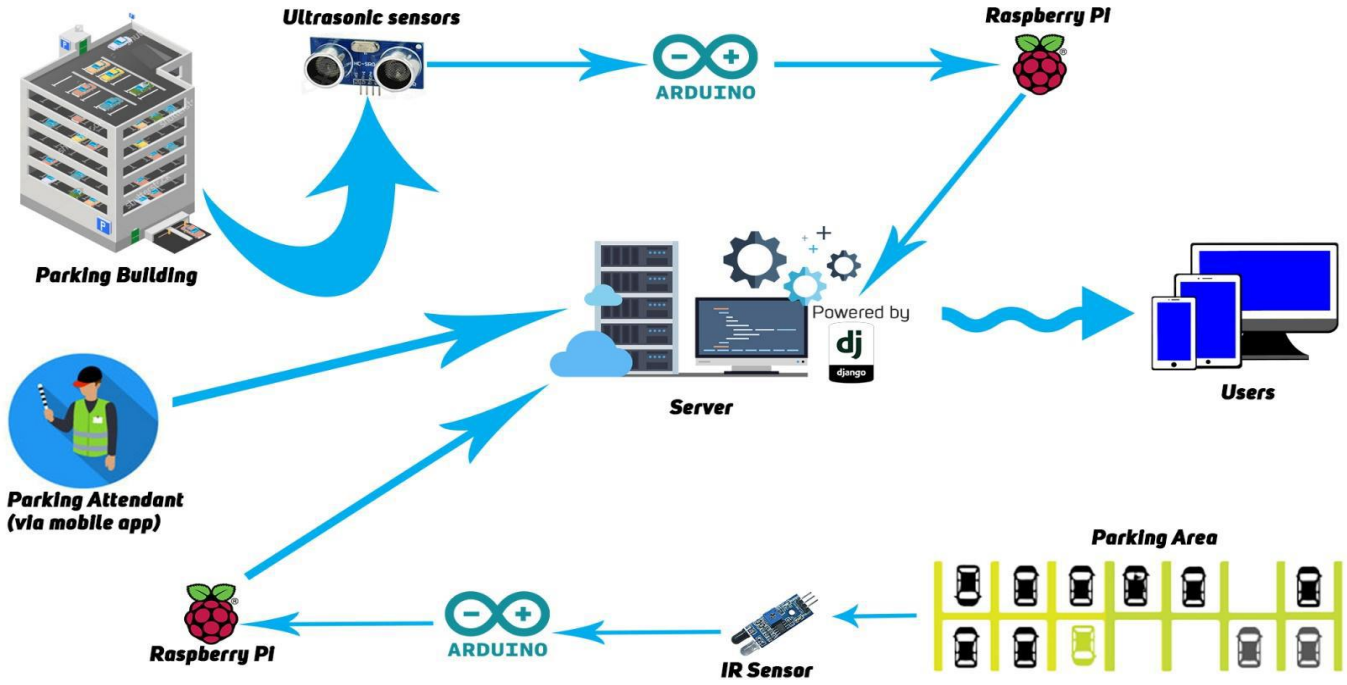
```
delay(1000);
Serial.begin (9600); // serial debugging
Wire.begin(D2, D1); // i2c start
myservo.attach(D6); // servo pin to D6
myservos.attach(D5); // servo pin to D5
pinMode(TRIG, OUTPUT); // trig pin as output
pinMode(ECHO, INPUT); // echo pin as input
pinMode(led, OUTPUT); // spot indication
pinMode(carExited, INPUT); // ir as input
pinMode(carEnter, INPUT); // ir as input
WiFi.begin(WIFI_SSID, WIFI_PASSWORD); //try to
connect with wifi
Serial.print("Connecting to ");
Serial.print(WIFI_SSID); // display ssid
while (WiFi.status() != WL_CONNECTED) {
Serial.print("."); // if not connected print this
delay(500);
}
Serial.println();
Serial.print("Connected to ");
Serial.println(WIFI_SSID);
Serial.print("IP Address is : ");
Serial.println(WiFi.localIP()); //print local IP address
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); // begin firebase
authentication
lcd.begin(); //begin lcd
lcd.home();
lcd.setCursor(0, 0); // 0th row and 0thh column
lcd.print("Smart Parking");
}
void loop() {
digitalWrite(TRIG, LOW); // make trig pin low
delayMicroseconds(2);
11
digitalWrite(TRIG, HIGH); // make trig pin high
delayMicroseconds(10);
digitalWrite(TRIG, LOW);
duration = pulseIn(ECHO, HIGH);
distance = (duration / 2) / 29.1; // take distance in cm
Serial.print("Centimeter: ");
Serial.println(distance);
int carEntry = digitalRead(carEnter); // read ir input
if (carEntry == HIGH) { // if high then count and send data
countYes++; //increment count
Serial.print("Car Entered = " ); Serial.println(countYes );
lcd.setCursor(0, 1);
lcd.print("Car Entered");
for (pos = 140; pos >= 45; pos -= 1) { // change servo position
myservos.write(pos);
delay(5);
}
```

```
delay(2000);
for (pos = 45; pos <= 140; pos += 1) { // change servo position
// in steps of 1 degree
myservos.write(pos);
delay(5);
}
Firebase.pushString("/Parking Status/", fireAvailable ); // send string to
firebase
lcd.clear();
}
int carExit = digitalRead(carExited); //read exit ir sensor
if (carExit == HIGH) { //if high then count and send
countYes--; //decrement count
Serial.print("Car Exited = " ); Serial.println(countYes);
lcd.setCursor(0, 1);
lcd.print("Car Exited");
for (pos1 = 140; pos1 >= 45; pos1 -= 1) { // change servo position
myservo.write(pos1);
delay(5);
}
delay(2000);
for (pos1 = 45; pos1 <= 140; pos1 += 1) { // change servo position
// in steps of 1 degree
myservo.write(pos1);
delay(5);
}
Firebase.pushString("/Parking Status/", fireAvailable ); // send string to firebase
lcd.clear();
}
if (distance < 6) { //if distance is less than 6cm then on led
Serial.println("Occupied ");
digitalWrite(led, HIGH);
}
if (distance > 6) { //if distance is greater than 6cm then off led
Serial.println("Available ");
digitalWrite(led, LOW);
}
Empty = allSpace - countYes; //calculate available data
Available = String("Available= ") + String(Empty) + String("/") +
String(allSpace); // convert the int to string
fireAvailable = String("Available=") + String(Empty) + String("/") +
String(allSpace);
lcd.setCursor(0, 0);
lcd.print(Available); //print available data to lcd
}
```

SCRIPT OF RASPBERRY PI:

Here's a simplified Python script for the Raspberry Pi to send data to the mobile app:

```
import paho.mqtt.client as mqtt
import time

# MQTT settings
MQTT_BROKER = "your_mqtt_broker_address"
MQTT_PORT = 1883
MQTT_TOPIC = "your_mqtt_topic"

# Initialize MQTT client
client = mqtt.Client()

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker")
    else:
        print(f"Connection failed with code {rc}")

client.on_connect = on_connect
client.connect(MQTT_BROKER, MQTT_PORT, 60)

try:
    while True:
        # Your data collection logic here
        data = "Your parking availability data goes here"

        # Send data to the mobile app
        client.publish(MQTT_TOPIC, data)

        print(f"Data sent: {data}")
        time.sleep(10)  # Adjust the interval as needed

except KeyboardInterrupt:
    print("Script terminated by user")
    client.disconnect()
```



FRAMEWORK IN FLUTTER:

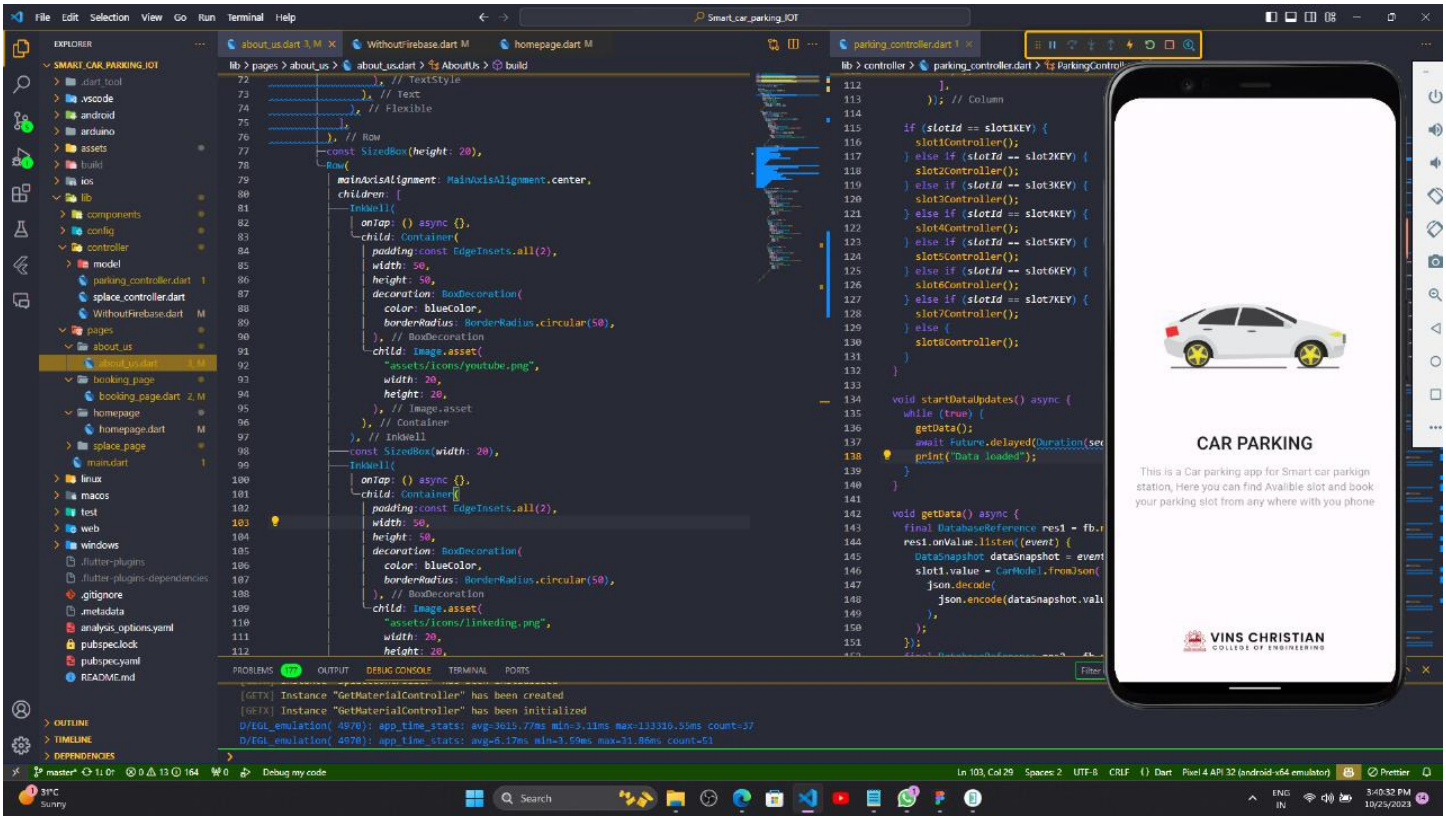


Fig:1

Fig1:Design the app's user interface, including screens, widgets, and layouts. Consider using Flutter's extensive library of pre-built widgets.

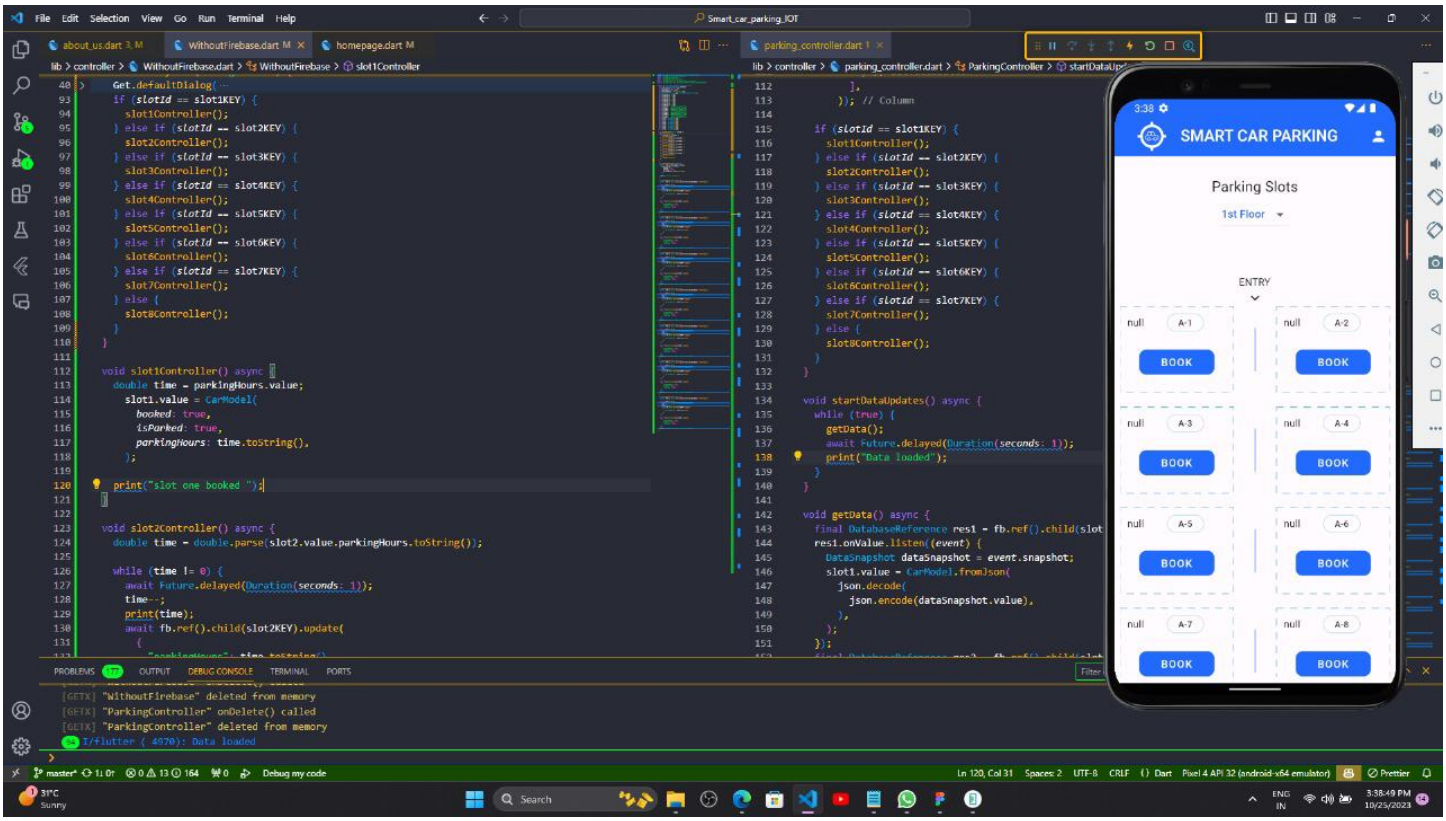
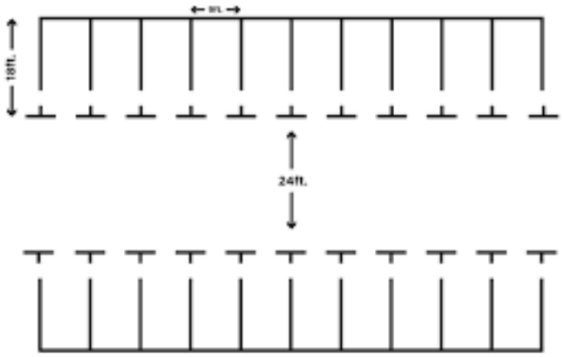


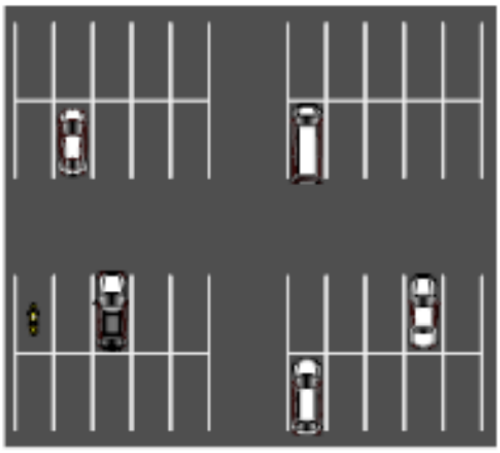
FIG :2

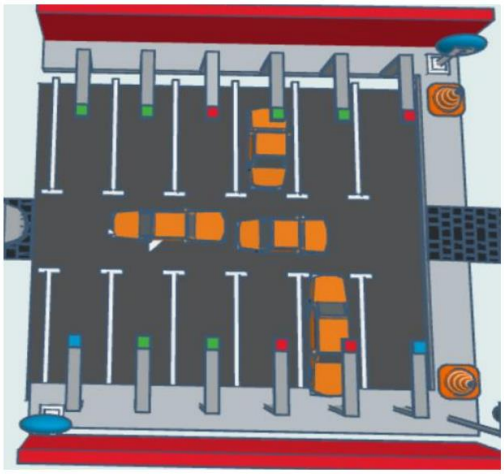
Fig2: The Design of the Mobile APP For SMART PARKING frame work has been successfully Completed..Kindly refer above fluter code and pictures.

DIAGRAM:

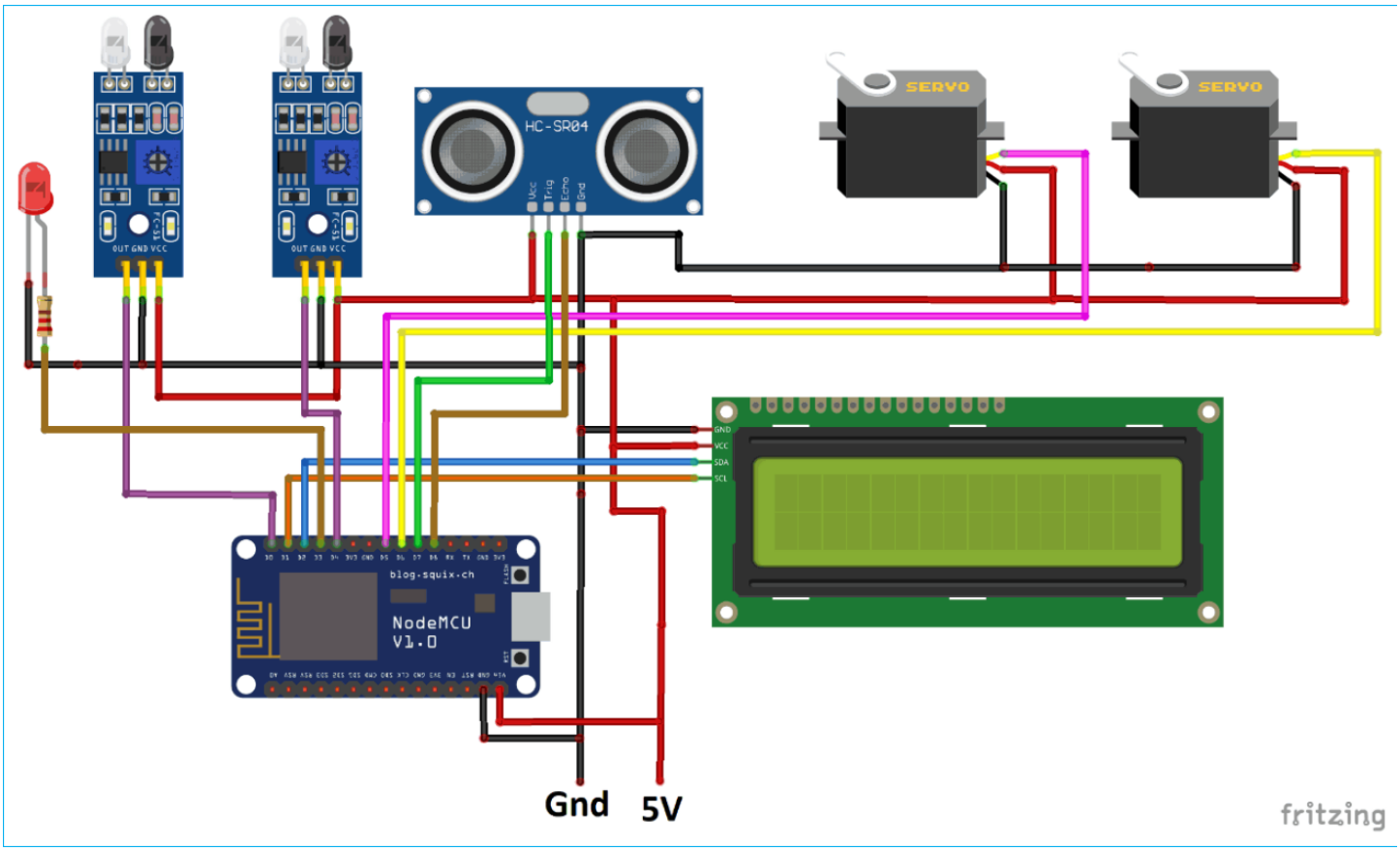


SCHEMATIC:

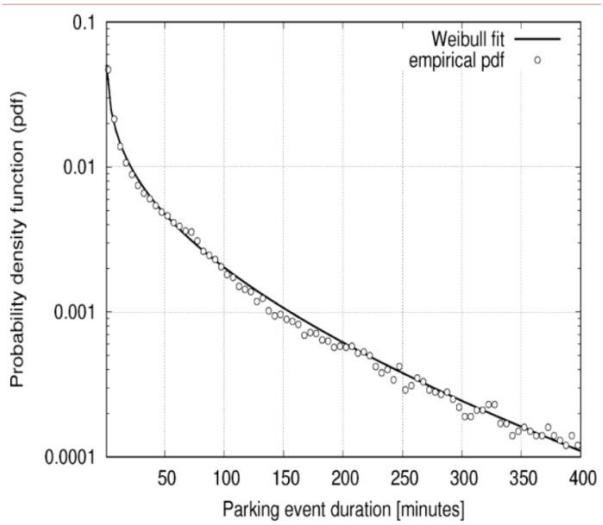




IOT DEVICE's:



DATA SHARING:



WORKING OF(SMART PARKING) APP:

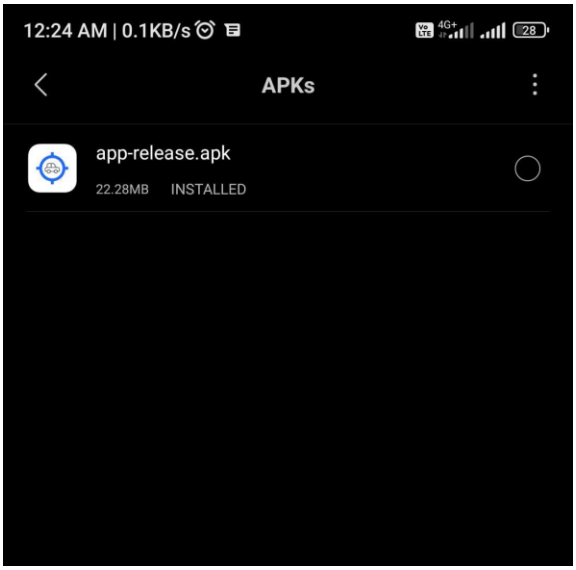


FIG:3 APP LOGO(smart_car_parking)

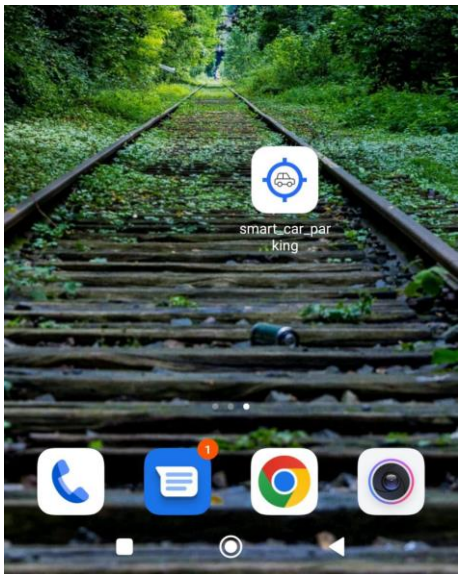


FIG:4 APP INSTALLED IN MO

PROCESS IN MOBILE APP WITH SCEENSHOT:

FIG:5 FRONT PAGE OF APP

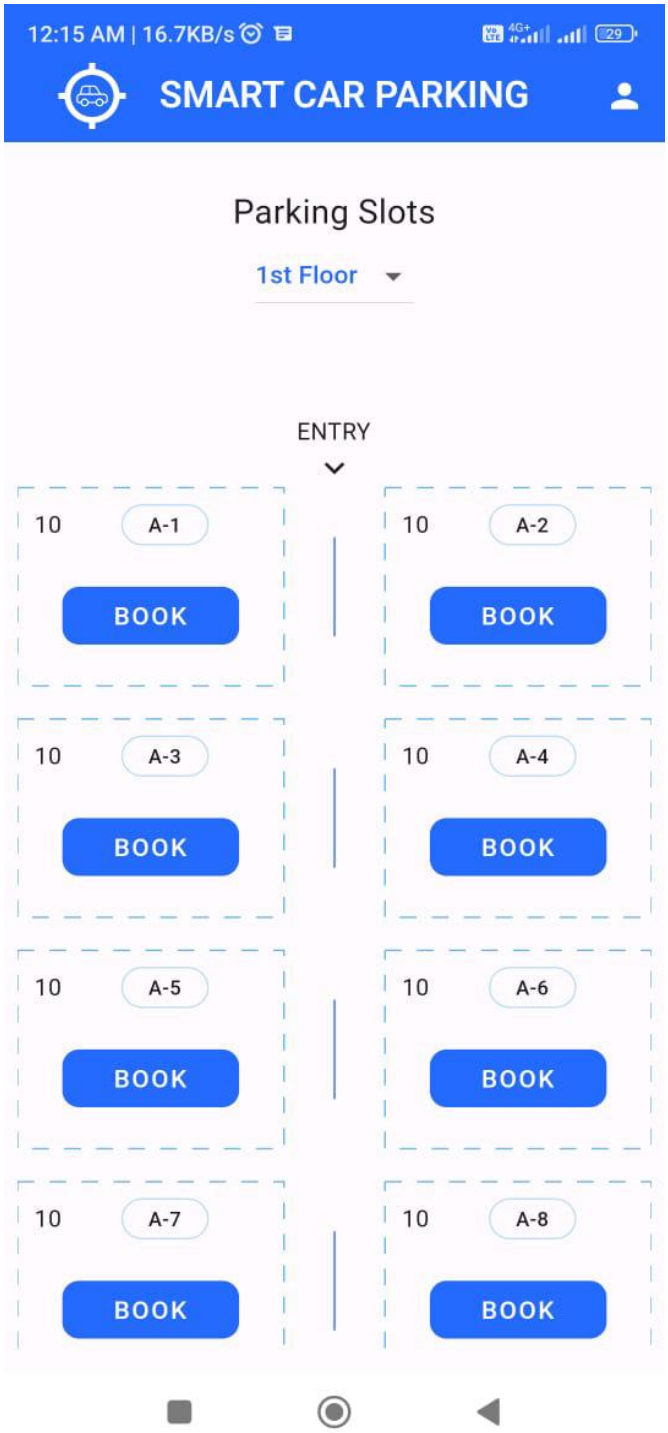
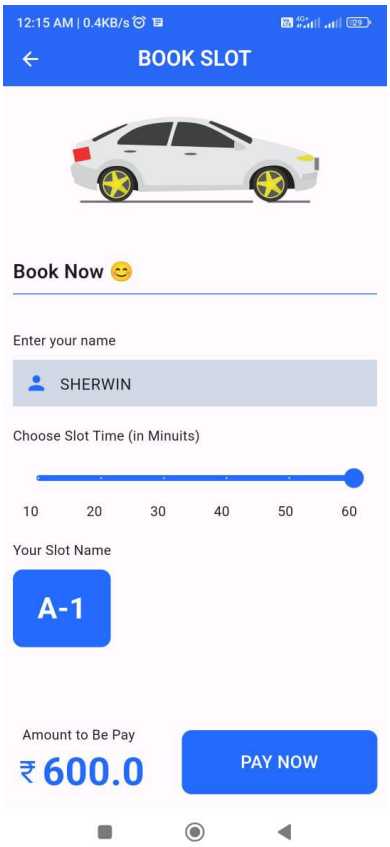


FIG:6 FREE SPACE CAN BE BOOKED



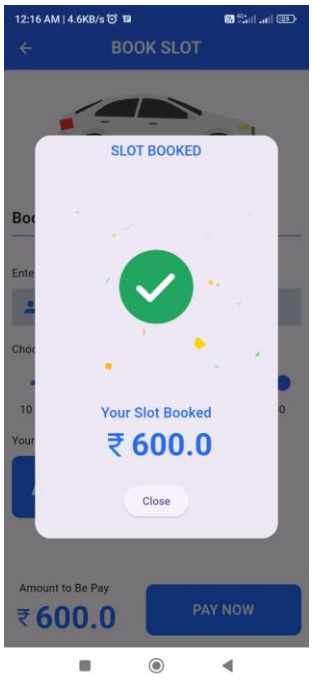


FIG:7,8 PARKING SPACE BOOKING

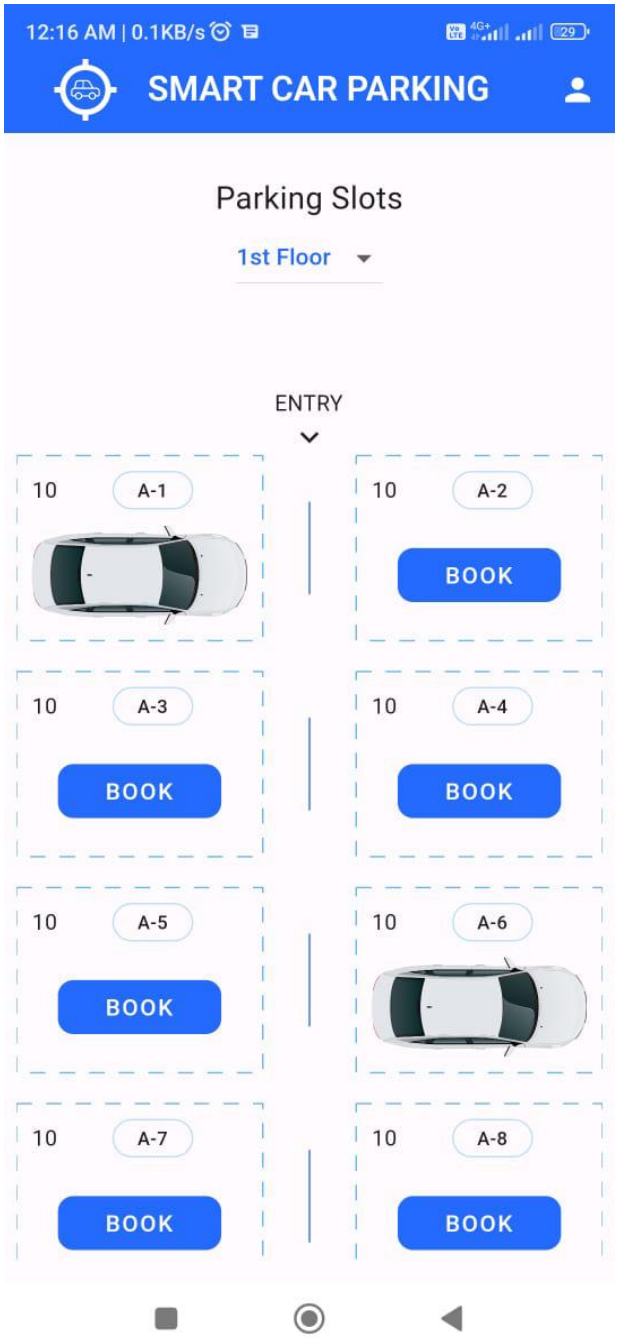


FIG:9 IT REPRESENT THE BOOKED SPACE

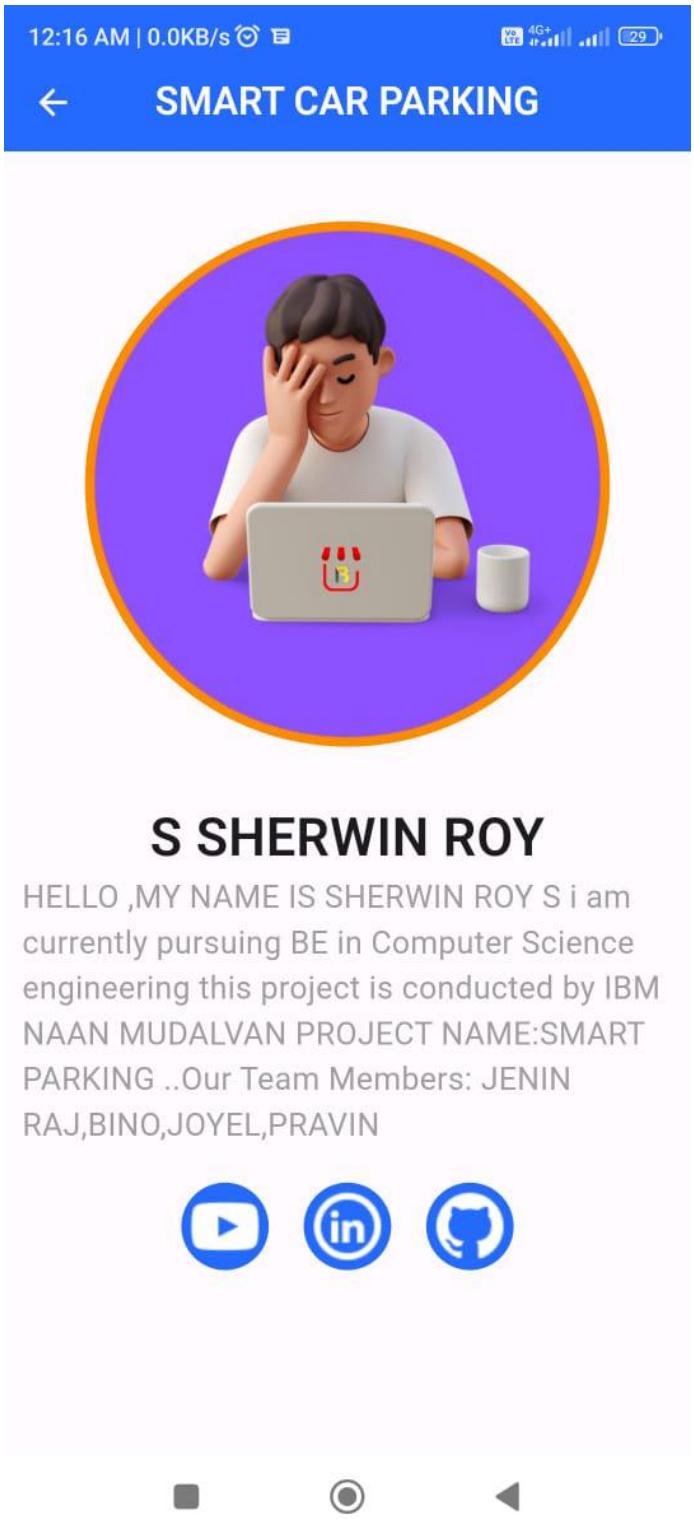


FIG:10 IT SHOWS THE MOBILE APP DASHBOARD

BENEFIT’s OF SMART PARKING:

Smart parking in IoT offers several benefits, including:

Efficiency: Smart parking systems provide real-time information about parking space availability, helping drivers find open spots quickly. This reduces the time spent searching for parking, alleviating traffic congestion and improving overall traffic flow.

Convenience: Users can access parking availability information through mobile apps or digital signs, making it convenient to plan their parking in advance. This can lead to a stress-free parking experience.

Reduced Emissions: By reducing the time spent circling for parking, smart parking systems can lower vehicle emissions and contribute to improved air quality in urban areas.

Optimized Space Usage: Smart parking systems help operators maximize the utilization of parking spaces, ensuring that spaces are not left empty and are used more efficiently.

Revenue Generation: Cities and parking operators can generate additional revenue by offering premium parking services, such as reservations, dynamic pricing, and advertising on digital displays.

Data Insights: Smart parking systems collect valuable data on parking patterns, which can be used for urban planning, optimizing parking resources, and predicting parking demand.

Improved User Experience: Drivers benefit from a more seamless parking experience, reducing frustration and stress associated with finding parking spaces.

Safety: Some smart parking systems incorporate security features, such as surveillance cameras, enhancing the safety of parking facilities.

Accessibility: Smart parking solutions can include features for individuals with disabilities, ensuring that accessible parking spaces are available and well-maintained.

Sustainability: By reducing unnecessary driving in search of parking, smart systems contribute to lower fuel consumption and a reduced carbon footprint.

In summary, smart parking systems in IoT significantly improve the efficiency, convenience, and sustainability of urban parking. They offer benefits to both drivers

and city planners, making urban mobility more manageable and environmentally friendly.

PROJECT IN DETAIL:

1. Project Overview:

   • Start with an executive summary that provides a brief description of the smart parking project. Explain the purpose and expected benefits of the system, such as reduced congestion, enhanced user experience, or increased revenue generation.

2. Project Objectives:
   • Clearly define the goals and objectives of the project. Be specific about what you aim to achieve, such as the number of parking spaces to be monitored, the reduction in search time, or the revenue targets.

3. Scope and Features:

   • Outline the scope of the project. List the key features and functionalities of the smart parking system. This may include real-time space monitoring, reservation and payment systems, user apps, security features, environmental considerations, and more.

4. Stakeholders:

   • Identify the project stakeholders, such as city authorities, property owners, users, and technology partners. Define their roles and responsibilities in the project.

5. Hardware and Sensors:

   • Specify the hardware components and sensors to be used. Detail the types of sensors (e.g., ultrasonic, infrared, camera-based), microcontrollers or platforms (e.g., ESP8266, Raspberry Pi), communication modules, and any displays or signage.

6. Network Infrastructure:

   • Describe the network and communication infrastructure. Specify whether you'll use Wi-Fi, LoRa, cellular networks, or a combination to connect sensors to the central server or cloud platform.

7. Central Server or Cloud Platform:

   • Provide details on the central server or cloud platform used to collect, process, and store data. Mention the database system and technologies you'll use.

8. User Interfaces:

- Explain the user interfaces, including mobile apps for users to find and reserve parking spaces, make payments, and receive real-time updates. Also, describe any web-based dashboards for administrators.

9. Payment Integration:

- Specify the payment gateways and options for users, such as cashless payments, credit card payments, and mobile wallets.

10. Data Analytics and Prediction:

- Outline how you plan to use data analytics for optimizing parking, pricing, and improving operational efficiency. Describe any machine learning or predictive analytics techniques.

11. Security and Access Control:

- Describe security measures to protect data and transactions. Include user authentication and access control mechanisms.

12. Environmental Considerations:

- Detail any sustainable features like electric vehicle charging stations, solar power, or green infrastructure.

13. Regulatory Compliance:

- Explain how your system will comply with local parking regulations, privacy laws, and other relevant regulations.

14. System Testing and Maintenance:

- Define a testing strategy to ensure system reliability and accuracy. Plan for regular maintenance of sensors and the entire system.

15. User Education and Onboarding:

- Outline how you will educate users on using the platform and ensure a smooth transition to smart parking.

16. Marketing and Adoption:

- Describe marketing strategies to promote the platform and encourage user adoption. Include partnerships and marketing campaigns.

17. Project Timeline:

• Create a detailed project timeline that specifies milestones, deadlines, and dependencies between tasks.

18. Budget and Resources:

• Estimate the budget required for the project, including hardware costs, software development, personnel, and ongoing operational expenses.

19. Risk Management:

• Identify potential risks and challenges in the project, along with mitigation strategies.

20. Monitoring and Evaluation:

• Explain how you will continuously monitor the system's performance and gather user feedback for ongoing improvement.

CONCLUSION:

• Summarize the project plan and restate the expected benefits and outcomes of the smart parking system.

This detailed project plan provides a roadmap for the development, implementation, and management of a smart parking system. It serves as a comprehensive guide for all stakeholders involved in the project.

LINKS:

GIT HUB LINK:
https://github.com/PravinV002/PravinV002/tree/main

APP LINK:{IN GITHUB}
https://github.com/PravinV002/PravinV002/tree/main/IOT_Phase4/APP

APP WORKING VIDEO LINK:

https://www.youtube.com/shorts/_5Xx0uObm3g

PHASE1 LINK:

https://github.com/PravinV002/PravinV002/tree/main/Iot_phase1

PHASE2 LINK:

https://github.com/PravinV002/PravinV002/tree/main/IOT_Phase2

PHASE3 LINK:

https://github.com/PravinV002/PravinV002/tree/main/IOT_Phase3

PHASE4 LINK:

https://github.com/PravinV002/PravinV002/tree/main/IOT_Phase4

PHASE5 LINK:

https://github.com/PravinV002/PravinV002/tree/main/IOT_Phase5

THANKING YOU