INTERNET OF THINGS

SMART_PARKING

Phase 4 Project Submission

Introduction:

In this phase of the project, you will focus on building a mobile app using Python and React Native. This app's primary purpose is to display real-time parking availability data received from a Raspberry Pi. You will design the app functions to efficiently receive and present this data to the users.

Source Code Example:

Here's an example of how you can start building your React Native mobile app to display parking availability data. Please note that this is a simplified example, and you'll need to adapt it to your specific requirements and integrate it with your Python code that communicates with the Raspberry Pi.

1. **Install React Native:**
   First, make sure you have React Native set up on your development environment. You can follow the official React Native documentation for installation instructions: https://reactnative.dev/docs/environment-setup

2. **Create a New React Native Project**

   Use the following command to create a new React Native project:

   ```
   npx react-native init ParkingAvailabilityApp
   ```

3. Design the User Interface:

Create components and screens to display parking availability data. You can use libraries like react-navigation for navigation between screens and react-native-elements for UI components.

4.Fetch and Display Data:

Use the fetch API or a library like axios to make HTTP requests to your Python server running on the Raspberry Pi.

Here's a simplified example of fetching data and displaying it:

```
import React, { useState, useEffect } from 'react';
import { View, Text } from 'react-native';
function ParkingAvailabilityScreen() {
 const [availabilityData, setAvailabilityData] = useState(null);
 useEffect(() => {
  // Make an HTTP request to your Python server (replace with your server URL)
  fetch('http://your-raspberrypi-server:port/availability')
   .then((response) => response.json())
   .then((data) => setAvailabilityData(data))
   .catch((error) => console.error(error));
 }, []);
 return (
  <View>
   {availabilityData ? (
    <Text>Parking Availability: {availabilityData.available}</Text>
   ) : (
```

```
        <Text>Loading data...</Text>
      )}
    </View>
  );
}
export default ParkingAvailabilityScreen;
```

5.Run the App:

Use the following commands to run your React Native app on connected device or emulator:

```
npx react-native start
npx react-native run-android    # For Android
npx react-native run-ios      # For iOS
```

6. Further Integration:

Your Python code running on the Raspberry Pi should expose an API endpoint that provides real-time parking availability data. Make sure to replace the URL in the fetch request with the appropriate endpoint from your Python server.

Remember that this is a basic example, and in a real-world scenario, you would want to handle error cases, optimize data fetching, and implement user-friendly UI. Additionally, security considerations should be taken into account when connecting to your Raspberry Pi.

By following this example and customizing it to your project's needs, you can create a mobile app using React Native to display real-time parking availability data from your Raspberry Pi server.

Kivy Framework For The Mobile App

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
```

```
from kivy.uix.label import Label
from kivy.uix.button import Button
from kivy.network.urlrequest import UrlRequest
import json
class ParkingAvailabilityApp(App):
    def build(self):
        self.layout = BoxLayout(orientation='vertical')
        self.availability_label = Label(text="Parking Availability: Loading...")
        self.update_button = Button(text="Refresh Data")
        self.update_button.bind(on_press=self.update_availability)
        self.layout.add_widget(self.availability_label)
        self.layout.add_widget(self.update_button)
        return self.layout
    def update_availability(self, instance):
        url = "https://your-api-endpoint.com/parking-availability"
        UrlRequest(url, self.parse_availability)
    def parse_availability(self, request, result):
        try:
            data = json.loads(result)
            available = data.get("available")
            last_updated = data.get("lastUpdated")
            self.availability_label.text = f"Parking Availability: {available} spots
available\nLast updated: {last_updated}"
        except Exception as e:
            self.availability_label.text = "Failed to fetch data"
if __name__ == '__main__':
    ParkingAvailabilityApp().run()
```

Python Code :

```
import time
Vehicle_Number=['XXXX-XX-XXXX']
Vehicle_Type=['Bike']
vehicle_Name=['Intruder']
Owner_Name=['Unknown']
Date=['22-22-3636']
Time=['22:22:22']
bikes=100
cars=250
bicycles=78
def main():
    global bikes,cars,bicycles
    try:
        while True:
            print("------------------------------------------------------------------------")
            print("\t\tParking Management System")
            print("------------------------------------------------------------------------")
            print("1.Vehicle Entry")
            print("2.Remove Entry" )
            print("3.View Parked Vehicle ")
            print("4.View Left Parking Space ")
            print("5.Amount Details ")
            print("6.Bill")
            print("7.Close Programme ")
            print("+---------------------------------------+")
            ch=int(input("\tSelect option:"))
            if ch==1:
                no=True
```

```
while no==True:
    Vno=input("\tEnter vehicle number (XXXX-XX-XXXX) - ").upper()
    if Vno=="":
        print("###### Enter Vehicle No. ######")
    elif Vno in Vehicle_Number:
        print("###### Vehicle Number Already Exists")
    elif len(Vno)==12:
        no=not True
        Vehicle_Number.append(Vno)
    else:
        print("###### Enter Valid Vehicle Number ######")
typee=True
while typee==True:
    Vtype=str(input("\tEnter vehicle type(Bicycle=A/Bike=B/Car=C):")).lower()
    if Vtype=="":
        print("###### Enter Vehicle Type ######")
    elif Vtype=="a":
        Vehicle_Type.append("Bicycle")
        bicycles-=1
        typee=not True
    elif Vtype=="b":
        Vehicle_Type.append("Bike")
        bikes-=1
        typee=not True
    elif Vtype=="c":
        Vehicle_Type.append("Car")
        cars-=1
        typee=not True
    else:
        print("###### Please Enter Valid Option ######")
```

```
name=True
while name==True:
    vname=input("\tEnter vehicle name - ")
    if vname=="":
        print("########Please Enter Vehicle Name ########")
    else:
        vehicle_Name.append(vname)
        name=not True
o=True
while o==True:
    OName=input("\tEnter owner name - ")
    if OName=="":
        print("###### Please Enter Owner Name ######")
    else:
        Owner_Name.append(OName)
        o=not True
d=True
while d==True:
    date=input("\tEnter Date (DD-MM-YYYY) - ")
    if date=="":
        print("###### Enter Date ######")
    elif len(date)!=10:
        print("###### Enter Valid Date ######")
    else:
        Date.append(date)
        d=not True
t=True
while t==True:
    time=input("\tEnter Time (HH:MM:SS) - ")
    if t=="":
```

```
            print("###### Enter Time ######")
        elif len(time)!=8:
            print("###### Please Enter Valid Date ######")
        else:
            Time.append(time)
            t=not True
    print("\n.....Record detail saved.....")
elif ch==2:
    no=True
    while no==True:
        Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()
        if Vno=="":
            print("###### Enter Vehicle No. ######")
        elif len(Vno)==12:
            if Vno in Vehicle_Number:
                i=Vehicle_Number.index(Vno)
                Vehicle_Number.pop(i)
                Vehicle_Type.pop(i)
                vehicle_Name.pop(i)
                Owner_Name.pop(i)
                Date.pop(i)
                Time.pop(i)
                no=not True
                print("\n.....Removed Sucessfully.....")
            elif Vno not in Vehicle_Number:
                print("###### No Such Entry ######")
            else:
                print("Error")
        else:
            print("###### Enter Valid Vehicle Number ######")
```

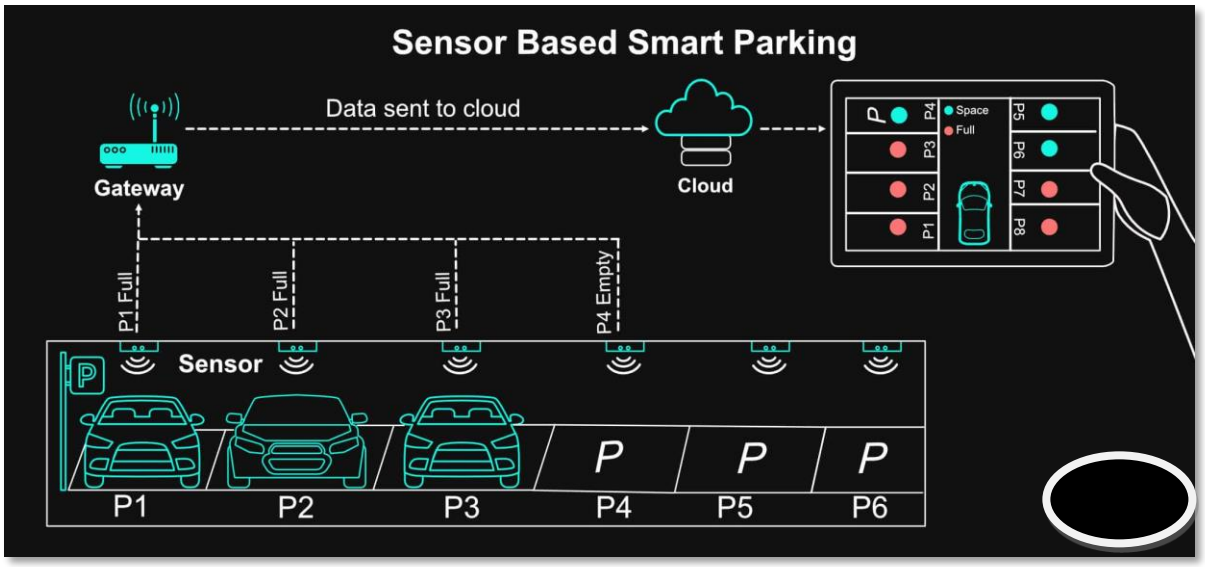
```
elif ch==3:
    count=0
    print("-------------------------------------------------------------------------------")
    print("\t\t\t\tParked Vehicle")
    print("-------------------------------------------------------------------------------")
    print("Vehicle No.\tVehicle Type  Vehicle Name\t  Owner Name\t Date\t \tTime")
    print("-------------------------------------------------------------------------------")
    for i in range(len(Vehicle_Number)):
        count+=1
       print(Vehicle_Number[i],"\t  ",Vehicle_Type[i],"\t ",vehicle_Name[i],"\t
       ",Owner_Name[i]," " ,Date[i]," ",Time[i])
    print("-------------------------------------------------------------------------------")
    print("---------------------- Total Records - ",count,"----------------------------------")
    print("-------------------------------------------------------------------------------")
elif ch==4:
    print("-------------------------------------------------------------------------------")
    print("\t\t\t\tSpaces Left For Parking")
    print("-------------------------------------------------------------------------------")
    print("\tSpaces Available for Bicycle - ",bicycles)
    print("\tSpaces Available for Bike - ",bikes)
    print("\tSpaces Available for Car - ",cars)
    print("-------------------------------------------------------------------------------")
elif ch==5:
    print("-------------------------------------------------------------------------------")
    print("\t\t\t\tParking Rate")
    print("-------------------------------------------------------------------------------")
    print("*1.Bicycle      Rs20 / Hour")
    print("*2.Bike         Rs40/ Hour")
    print("*3.Car          Rs60/ Hour")
    print("-------------------------------------------------------------------------------")
elif ch==6:
```

```
print("..... Generating Bill.....")
no=True
while no==True:
    Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()
    if Vno=="":
        print("###### Enter Vehicle No. ######")
    elif len(Vno)==12:
        if Vno in Vehicle_Number:
            i=Vehicle_Number.index(Vno)
            no=not True
        elif Vno not in Vehicle_Number:
            print("###### No Such Entry ######")
        else:
            print("Error")
    else:
        print("###### Enter Valid Vehicle Number ######")
print("\tVehicle Check in time - ",Time[i])
print("\tVehicle Check in Date - ",Date[i])
print("\tVehicle Type - ",Vehicle_Type[i])
inp=True
amt=0
while inp==True:
    hr=input("\tEnter No. of Hours Vehicle Parked - ").lower()
    if hr=="":
        print("###### Please Enter Hours ######")
    elif int(hr)==0 and Vehicle_Type[i]=="Bicycle":
        amt=20
        inp=not True
    elif int(hr)==0 and Vehicle_Type[i]=="Bike":
        amt=40
```

```
                inp=not True
            elif int(hr)==0 and Vehicle_Type[i]=="Car":
                amt=60
                inp=not True
            elif int(hr)>=1:
                if Vehicle_Type[i]=="Bicycle":
                    amt=int(hr)*int(20)
                    inp=not True
                elif Vehicle_Type[i]=="Bike":
                    amt=int(hr)*int(40)
                    inp=not True
                elif Vehicle_Type[i]=="Car":
                    amt=int(hr)*int(60)
                    inp=not True
            print("\t Parking Charge - ",amt)
            ac=18/100*int(amt)
            print("\tAdd. charge 18 % - ",ac)
            print("\tTotal Charge - ",int(amt)+int(ac))
            print(".....Thank you for using our service……")
            a=input("\tPress Any Key to Proceed - ")
        elif ch==7:
            print(".....Thank you for using our
service.....")
            print("                    **********(: Bye Bye :)**********")
            break
            quit
    except:
        main()
main()
```

Diagram :

Example output diagram:



Output for program:

Parking Management System

1.Vehicle Entry

2.Remove Entry

3.View Parked Vehicle

4.View Left Parking Space

5.Amount Details

6.Bill

7.Close Programme

+---+

Select option: 1

Enter vehicle number (XXXX-XX-XXXX) - KA-01-HH-1234

Enter vehicle type(Bicycle=A/Bike=B/Car=C): b

Enter vehicle name - Honda Activa

Enter owner name - John Doe

Enter Date (DD-MM-YYYY) - 20-08-2023

Enter Time (HH:MM:SS) - 14:30:00

.....Record detail saved.....

Select option: 3

Parked Vehicle

Vehicle No. Vehicle Type Vehicle Name Owner Name Date Time

KA-01-HH-1234 Bike Honda Activa John Doe 20-08-2023 14:30:00

---------- Total Records - 1 ----------

Select option: 6

..... Generating Bill .....

Enter vehicle number to Delete(XXXX-XX-XXXX) - KA-01-HH-1234

Vehicle Check in time - 14:30:00

Vehicle Check in Date - 20-08-2023

Vehicle Type - Bike

Enter No. of Hours Vehicle Parked - 1

Parking Charge - 40

Add. charge 18 % - 7.2

Total Charge - 47.2

.....Thank you for using our service.....

Press Any Key to Proceed

Conclusion:

the project discussed the development of a mobile app for displaying real-time parking space availability using Python and two different frameworks, React Native and Kivy.

React Native:

React Native is a widely used framework for building cross-platform mobile apps using JavaScript and React.

The provided React Native code demonstrated how to create a user-friendly mobile app that retrieves parking availability data from a Python server (Flask) and presents it to users in a clear and interactive manner.

React Native is a powerful choice for building professional, production-ready mobile apps that offer a native-like user experience on both iOS and Android platforms.

Kivy:

Kivy, another Python framework, allows for the creation of cross-platform mobile and desktop applications with graphical user interfaces.

The sample Kivy code showcased a basic mobile app that fetched parking availability data from an API and displayed it with a simple user interface, including a refresh button.

Kivy can be a practical option for smaller projects or when Python is the preferred language for both backend and frontend development.

The choice between React Native and Kivy depends on specific project requirements, scale, and personal preferences. React Native excels when you need a polished and professional mobile app with a wide user base, while Kivy is more suitable for smaller-scale or niche applications.

In both cases, these examples illustrated how Python can be utilized to build mobile applications, expanding the capabilities for real-time data presentation and interactivity on mobile devices. The key takeaway is that Python is a versatile language that can be applied to create mobile apps in different contexts and with varying levels of complexity.