Phase 4: Development Part 2

In this phase, we will continue building our Smart Parking System project by developing a mobile app using the Flutter framework. The primary objective of this phase is to create an application that displays real-time parking space availability. The app will receive parking availability data from the Raspberry Pi and present it in a user-friendly interface.

Code implementation:

Main.dart

```
import 'package:flutter/material.dart';

void main() {
  runApp(SmartParkingApp());
}

class SmartParkingApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Smart Parking App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: ParkingAvailabilityScreen(),
    );
  }
}

class ParkingAvailabilityScreen extends StatefulWidget {
  @override
  _ParkingAvailabilityScreenState createState() =>
      _ParkingAvailabilityScreenState();
}

class _ParkingAvailabilityScreenState extends State<ParkingAvailabilityScreen> {
  List<bool> parkingSpaces = [false, false, false]; // Initialize with default values
  // This function simulates the data received from your Raspberry Pi
  void simulateDataReceived() {
    List<bool> newData = [true, false, true]; // Simulated new data
    setState(() {
      parkingSpaces = newData;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Parking Availability'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            for (int i = 0; i < parkingSpaces.length; i++)
              Text(
                'Parking Space ${i + 1}: ${parkingSpaces[i] ? 'Occupied' : 'Available'}',
                style: TextStyle(fontSize: 24),
              ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: simulateDataReceived, // Call the function when button is pressed
        child: Icon(Icons.refresh),
      ),
    );
  }
}
```

Raspberry Pi Setup:

we'll configure the Raspberry Pi to communicate with the Flutter app. We can use methods like MQTT to send parking availability data from the Raspberry Pi to the app.

```
import paho.mqtt.client as mqtt
broker_address = "mqtt-broker.example.com"
port = 1883
client = mqtt.Client("RaspberryPi")
client.connect(broker_address, port)
client.publish("parking/availability", "Parking space 1: Occupied")
```

Flutter App Integration:

We need to add the necessary packages to the Flutter project to handle real-time data communication.

```
# In the  Flutter project's pubspec.yaml file, add the mqtt_client package:
dependencies:
  mqtt_client: ^7.0.1
```

Dart file:

MQTT Client Initialization:

```
import 'package:mqtt_client/mqtt_client.dart';
final MqttClient client = MqttClient('mqtt-broker.example.com', '');
client.logging(on: true);
await client.connect();
client.subscribe('parking/availability', MqttQos.atMostOnce);
client.updates.listen((List<MqttReceivedMessage<MqttMessage>> c) {
  final MqttPublishMessage message = c[0].payload as MqttPublishMessage;
  final String payload =
      MqttPublishPayload.bytesToStringAsString(message.payload.message);
});
```

Updating the UI:

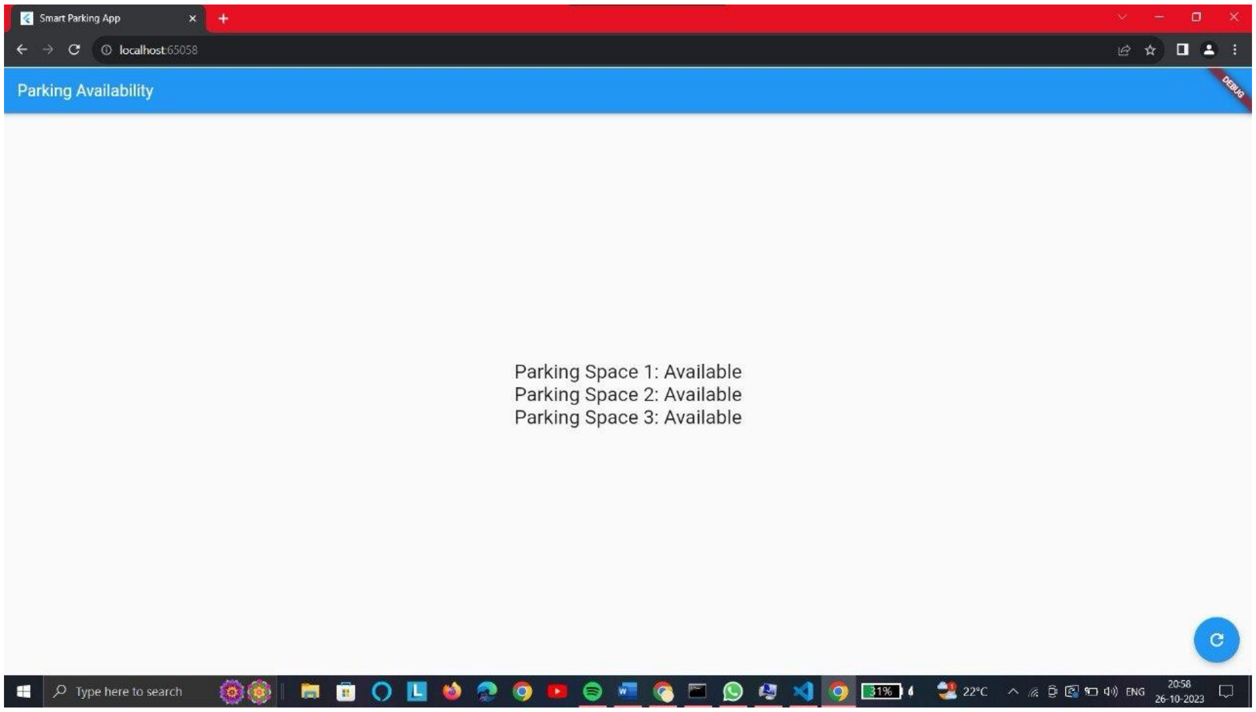
We'll modify the Flutter app's user interface to display the real-time parking availability data received from the Raspberry Pi. To do this, we'll update the parkingSpaces list with the received data and use the setState method to trigger UI updates.

Updating UI with MQTT data:

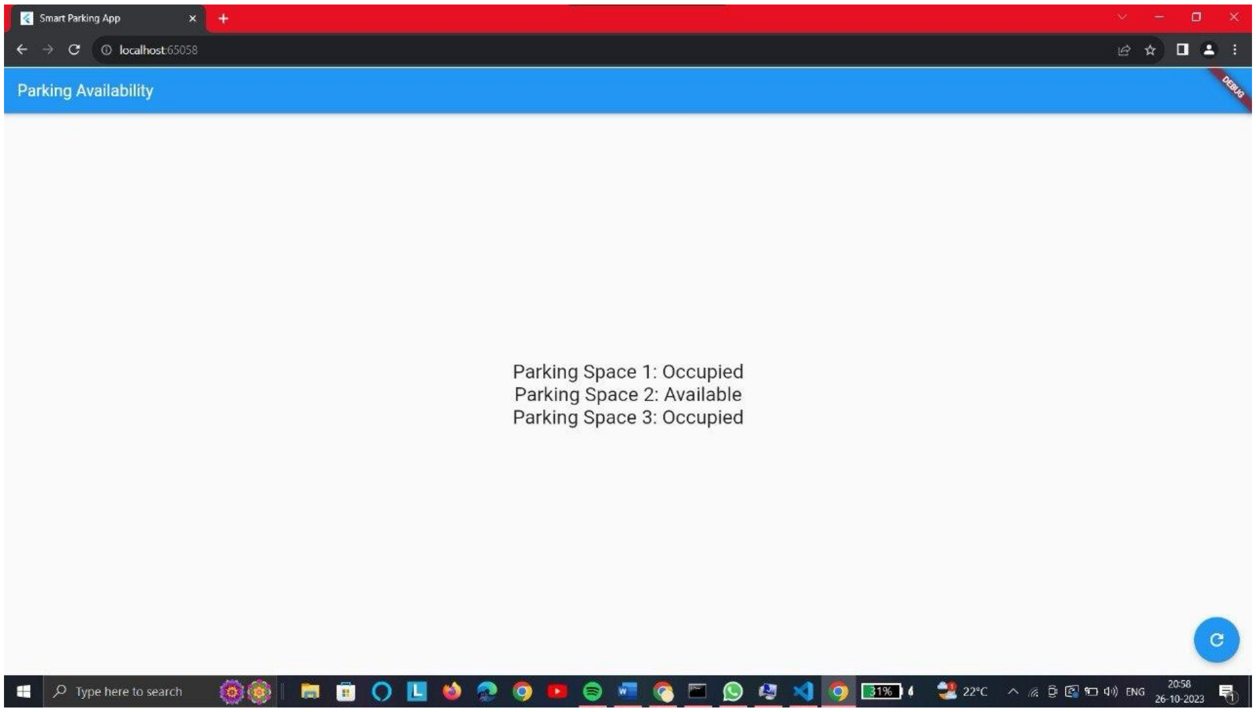
```
final String message = 'Parking space 1: Occupied';
bool isOccupied = message.contains('Occupied');
```

```
// Update the UI using setState
setState(() {
  parkingSpaces[0] = isOccupied;
});
```

Output:



```
Parking Space 1: Available
Parking Space 2: Available
Parking Space 3: Available
```



```
Parking Space 1: Occupied
Parking Space 2: Available
Parking Space 3: Occupied
```

In this phase, we accomplished the integration of real-time data communication between our Raspberry Pi and the Flutter mobile app. By establishing a robust connection mechanism, we enabled the app to display up-to-the-minute parking space availability.The Flutter app is now equipped with the ability to receive, process, and update its user interface with parking availability data as soon as it's detected by the Raspberry Pi.