**Phase:4**

To create a mobile app that displays real-time parking availability data received from a Raspberry Pi, we can use Flutter, a popular open-source UI software development kit created by Google. Flutter allows you to build natively compiled applications for mobile, web, and desktop from a single codebase. Here's how you can get started:

**Prerequisites:**

Install Flutter: Make sure you have Flutter and Dart installed. You can follow the official Flutter installation guide for your platform.

Development Environment: Set up your preferred code editor or IDE, such as Visual Studio Code with the Flutter and Dart plugins.

Raspberry Pi Server: Ensure that your Raspberry Pi is set up to collect and send parking availability data. You can use Python on the Raspberry Pi to create a server to provide this data.

**Steps to Create the Flutter Mobile App:**

**1. Create a New Flutter Project**:

You can create a new Flutter project using the Flutter CLI:

**flutter create parking\_availability\_app**

**cd parking\_availability\_app**

**2. Configure Dependencies**:

You'll need to configure the necessary dependencies for your app. Open the pubspec.yaml file in your project directory and add the following dependencies for real-time communication and UI components:

**dependencies:**

**flutter:**

**sdk: flutter**

**http: ^0.13.3 # For making HTTP requests**

**websocket: ^0.7.0 # For real-time communication**

**flutter\_map: ^0.13.0 # For displaying maps (if needed)**

**# Add other dependencies as per your requirements**

**Run flutter pub get to fetch these dependencies.**

**3. Design the App UI:**

Create the UI for your app, which should display parking availability data. You can design screens with widgets like ListView, Card, and Text to present the information.

**4. Implement Real-Time Communication:**

To receive real-time parking availability data, you can use websockets. You will need to connect to the Raspberry Pi server using a WebSocket library like websocket or any other preferred library. Implement WebSocket logic to listen for updates from the server.

Here's a simplified example of how to establish a WebSocket connection:

**import 'package:websocket/websocket.dart';**

**void connectToWebSocket() {**

**final socket = WebSocket('ws://raspberrypi\_ip:port');**

**socket.onOpen.listen((\_) {**

**print('WebSocket connected');**

**});**

**socket.onMessage.listen((data) {**

**// Handle parking availability data received from the server**

**});**

**socket.onClose.listen((\_) {**

**print('WebSocket disconnected');**

**});**

**}**

5**. Update UI with Real-Time Data:**

As you receive parking availability updates from the WebSocket, update your UI to display the latest data. You can use setState or state management solutions like Provider or Bloc to manage the app's state and trigger UI updates.

**6. Integrate Maps (Optional):**

If you want to display parking locations on a map, you can use packages like flutter\_map or google\_maps\_flutter to integrate maps into your app.

7**. Test and Debug:**

Test your app on both Android and iOS simulators or real devices to ensure it works correctly. Use Flutter's debugging tools for troubleshooting any issues.

8. **Build and Publish:**

Once your app is complete, you can build and package it for distribution on Google Play Store (Android) and the Apple App Store (iOS). Follow the respective platform-specific guidelines for app submission.