**Smart Parking(App development)**

**To create a mobile app that displays real-time parking availability data received from a Raspberry Pi using Python, you would typically use a mobile app development framework like Flutter, as mentioned. However, it's essential to clarify that Flutter primarily uses Dart as its programming language, not Python. If you want to use Python for mobile app development, you might consider frameworks like Kivy, BeeWare, or Pyqtdeploy.**

**But for this guide, I will continue with the assumption that you are using Flutter (with Dart). If you intend to use Python for mobile development, the process would differ significantly.**

**Here's how to design the app functions and create a model for real-time parking availability using Flutter:**

**Step 1: Create a Flutter Project:**

**If you haven't already created a Flutter project, follow the steps mentioned earlier to set up Flutter in Visual Studio Code and create a new project.**

**Step 2: Design the User Interface:**

**Design the app's user interface to display parking availability data. You might want to create screens that show a list of parking spots, their statuses, and any relevant information. Use Flutter widgets like `ListView`, `Card`, and `Text` to build your UI.**

**Step 3: Receive Data from Raspberry Pi:**

**To receive real-time parking availability data from your Raspberry Pi, you'll need to establish a communication channel between the app and the Raspberry Pi. You can use various methods, such as HTTP requests or WebSocket communication. For this example, I'll use a hypothetical HTTP API to receive data.**

**- Create a class or function to make HTTP requests to the Raspberry Pi's API. You can use the `http` package in Flutter to do this.**

**```dart**

**import 'package:http/http.dart' as http;**

**class ParkingApi {**

**static Future<Map<String, dynamic>> getParkingData() async {**

**final response = await http.get(Uri.parse('http://raspberry-pi-ip/api/parking'));**

**if (response.statusCode == 200) {**

**return jsonDecode(response.body);**

**} else {**

**throw Exception('Failed to load parking data');**

**}**

**}**

**}**

**```**

**Step 4: Model for Parking Data:**

**Create a model class to represent parking spot data. This class will help structure the data you receive from the Raspberry Pi.**

**```dart**

**class ParkingSpot {**

**final String name;**

**final bool isAvailable;**

**final String location;**

**ParkingSpot({**

**required this.name,**

**required this.isAvailable,**

**required this.location,**

**});**

**factory ParkingSpot.fromJson(Map<String, dynamic> json) {**

**return ParkingSpot(**

**name: json['name'],**

**isAvailable: json['isAvailable'],**

**location: json['location'],**

**);**

**}**

**}**

**```**

**Step 5: Data Display:**

**- Fetch the parking data from the Raspberry Pi's API using the `ParkingApi` class you created.**

**- Parse the data into a list of `ParkingSpot` objects.**

**- Display the parking availability information using Flutter widgets in your UI. You can use a `ListView.builder` to create a dynamic list of parking spots.**

**```dart**

**ListView.builder(**

**itemCount: parkingSpots.length,**

**itemBuilder: (context, index) {**

**final spot = parkingSpots[index];**

**return ListTile(**

**title: Text(spot.name),**

**subtitle: Text(spot.location),**

**trailing: spot.isAvailable ? Icon(Icons.check\_circle) : Icon(Icons.cancel),**

**);**

**},**

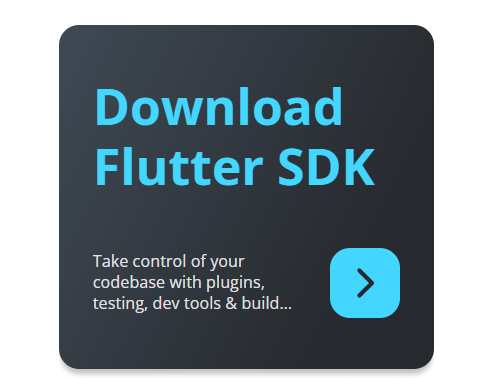
**)**

**```**

**Step 6: Real-Time Updates:**

**For real-time updates, consider implementing a mechanism like WebSockets or periodically polling the Raspberry Pi's API for new data. When new data is received, update your app's UI to reflect the changes.**

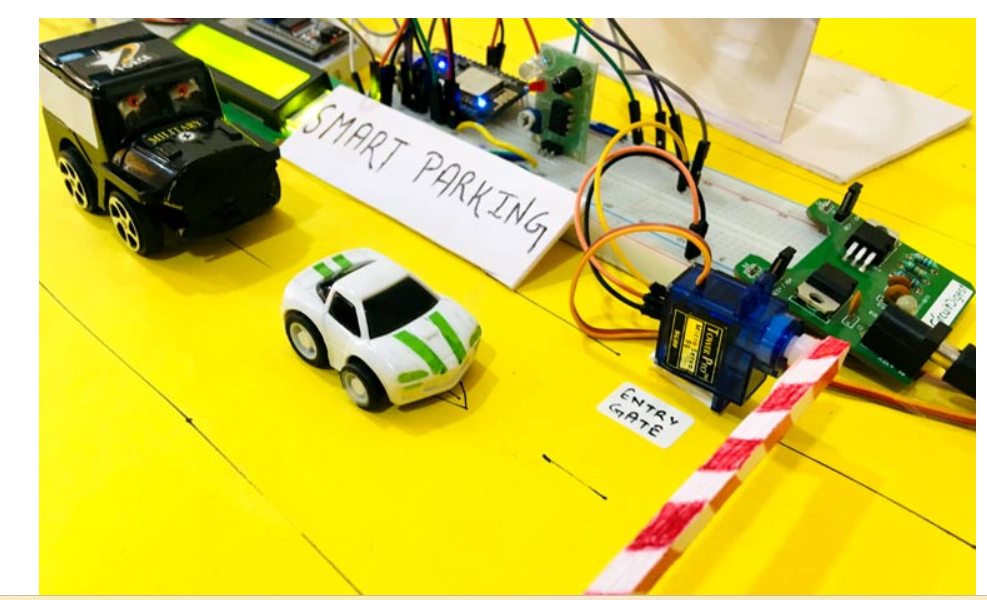
**Remember that this is a simplified example. In a real project, you would handle network errors, user authentication, and other considerations. Additionally, you'd likely use state management solutions like Provider or Bloc to manage your app's state and handle real-time updates efficiently.**

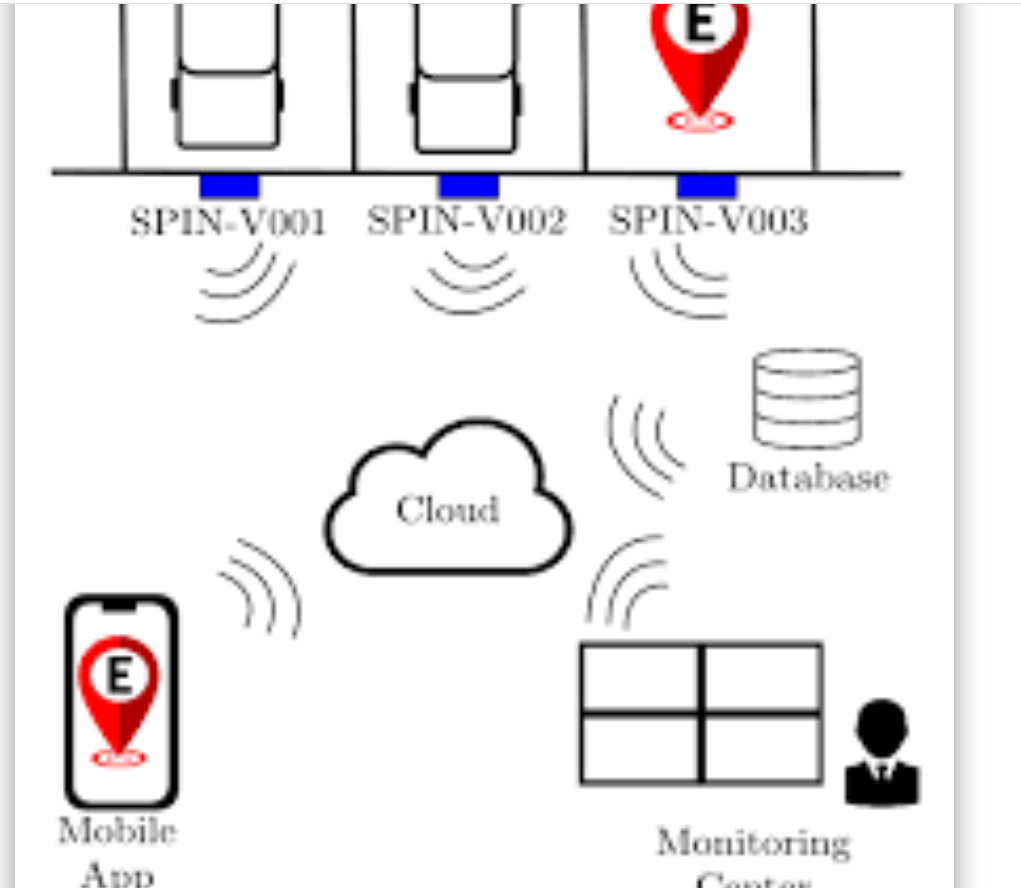
****

**A screenshot of a computer

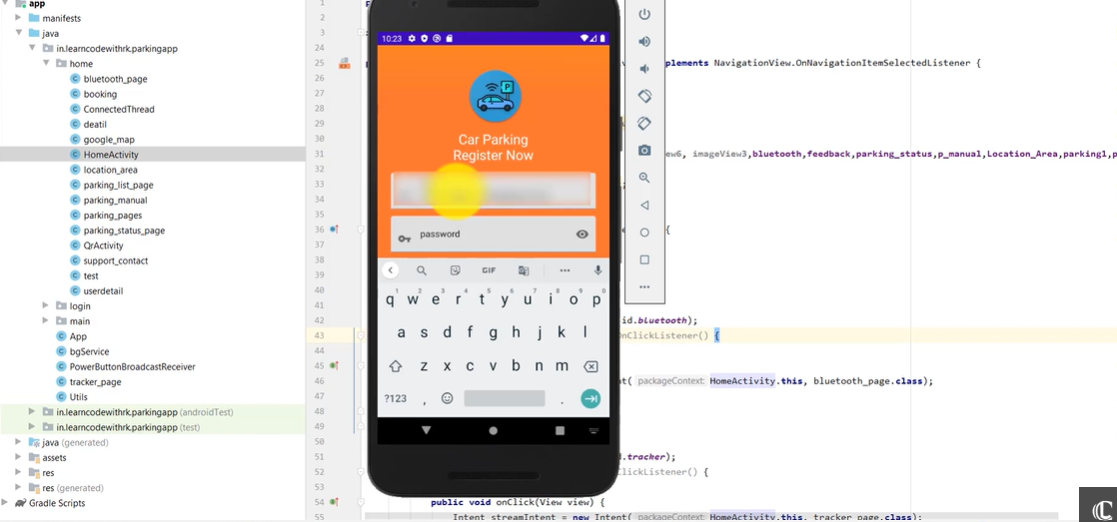
Description automatically generated**

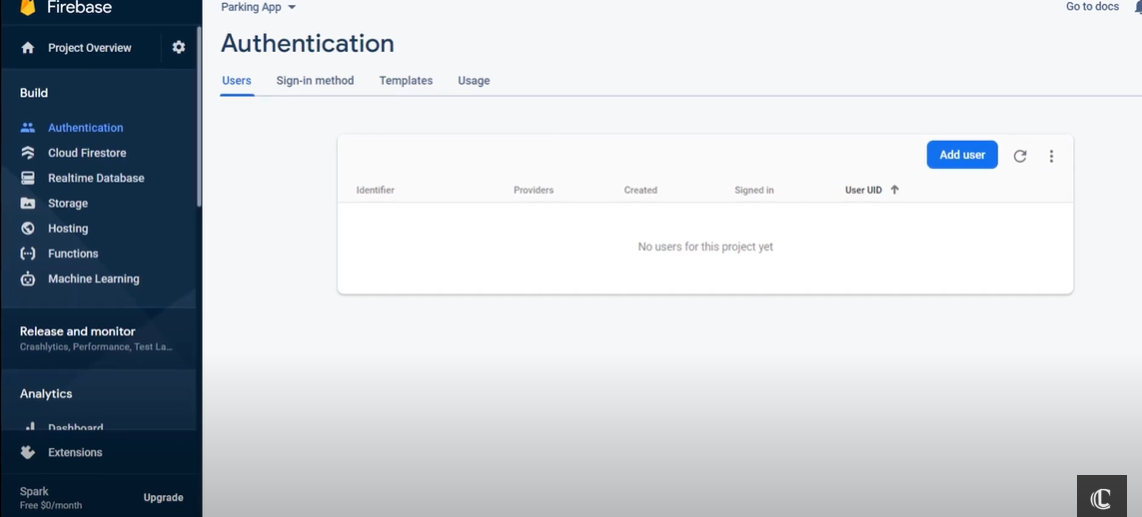
Model for smart parking:

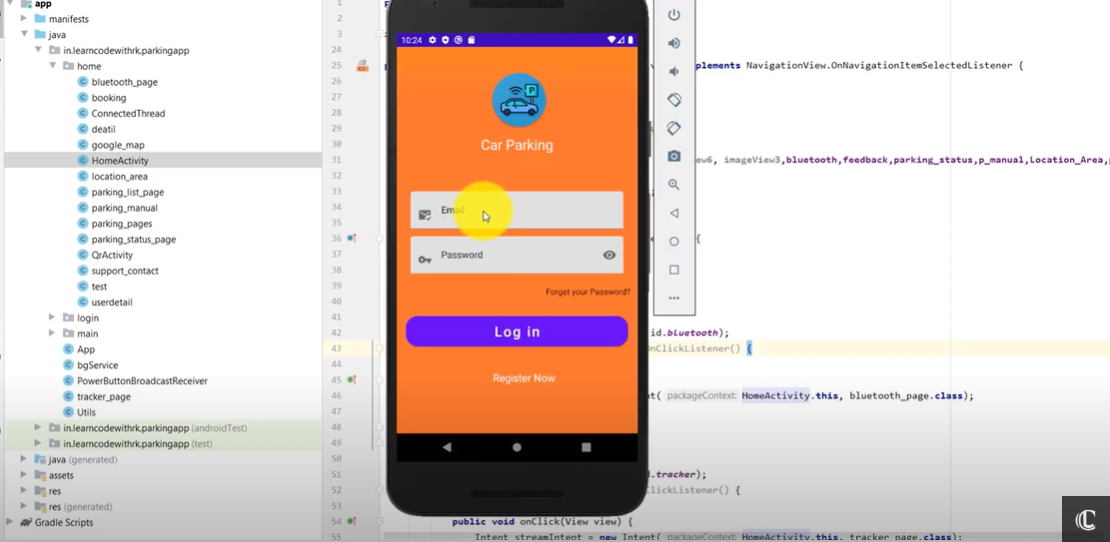


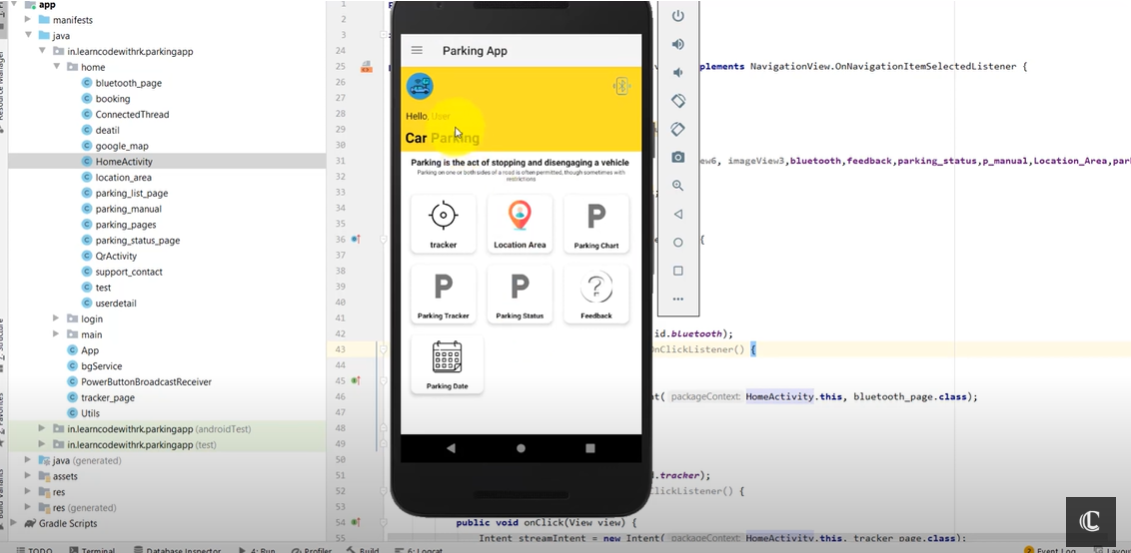
**Available space tracking:**

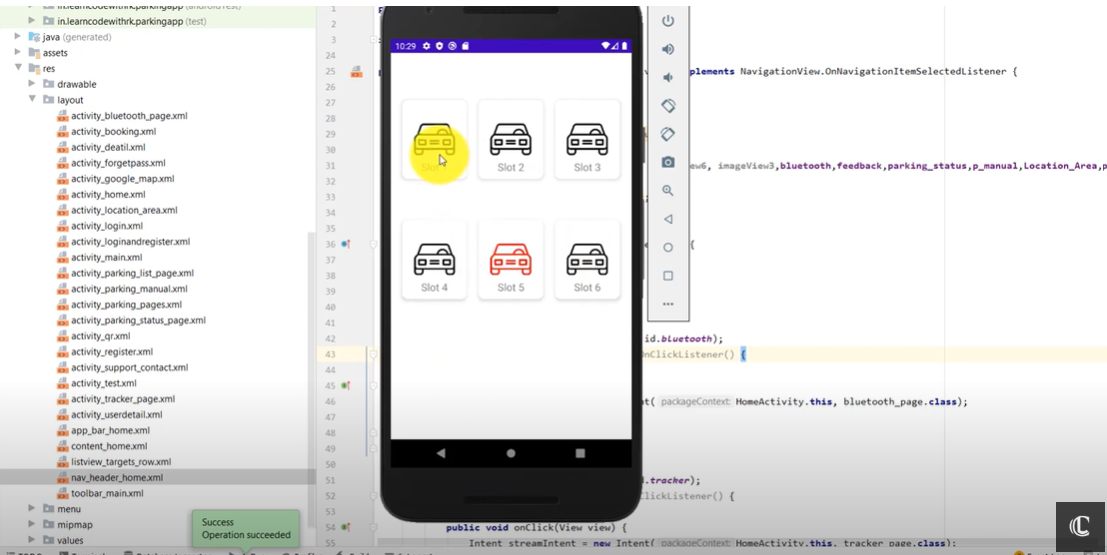
**Build a smart parking App:**



**Upload the firebase cloud** :







A cell phone with a graph on the screen

Description automatically generated