# SMART PARKING

In this part you will continue building your project. Continue building the project by developing the mobile app using python. Use a mobile app development framework (e.g., flutter) to create an app that displays realtime parking availability. Design app functions to receive and display parking availability data received from the raspberry pi.

## DEVELOPING THE MOBILE APP USING PYTHON:

While Python is a versatile programming language, it is not commonly used for mobile app development. However, there are frameworks and tools available that allow you to build mobile apps using Python. One such framework is Kivy.To build a smart parking project mobile app using Python and Kivy, you can follow these steps:

**1.Set up your development environment:**

Install Python and the necessary dependencies for Kivy development.

**2. Install Kivy:**

Use pip, the Python package manager, to install the Kivy framework.

**3. Design the user interface:**

Use Kivy's widget system to design the app's user interface. Define the layout, buttons, labels, and other elements required for your smart parking app.

**4. Integrate real-time data:**

Connect your app to a real-time parking availability data source. This could be an API provided by a parking management system or a third-party service that offers parking data. Use Python's networking capabilities to fetch and update the data.

**5. Implement real-time updates:**

If the parking availability data changes frequently, consider implementing a mechanism to receive real-time updates. This could involve using technologies like WebSockets or push notifications to keep the app's data up to date.

**6. Test and debug:**

Thoroughly test your app on different devices and simulate various scenarios to ensure its functionality and responsiveness. Debug any issues that arise during the testing phase.

**7. Package and distribute:**

Once you are satisfied with your app's performance, package it for distribution on Android and iOS devices. Kivy provides tools to package your Python app as a standalone executable or as an APK for Android devices.

## MOBILE APP DEVELOPMENT FRAMEWORK USING FLUTTER:

**1. Set up your development environment:**

- Install Python and Flutter on your computer.
- Set up the necessary dependencies for Flutter development.

**2. Create a new Flutter project:**

- Open a terminal or command prompt and run the command `flutter create parking_app`.
- Navigate to the newly created project directory using `cd parking_app`.

**3. Design the app UI:**

- Open the `lib/main.dart` file and update the default code with your own UI design.
- Add widgets to display the parking availability data received from the Esp32.

**4. Establish a connection with the Esp32:**

- Use a networking library like `http` to send HTTP requests to the Esp32 to fetch the parking availability data.
- Parse the response data and update the UI accordingly.

**5. Implement real-time updates:**

- Use a WebSocket library like `web_socket_channel` to establish a WebSocket connection with the Esp32.
- Listen for real-time updates on parking availability and update the app UI accordingly.

**6. Build and test the app:**

- Run the command `flutter run` in the terminal to build and launch the app on a connected device or emulator.
- Test the app by checking if it successfully displays the parking availability data and updates in real-time.

## RECEIVE AND DISPLAY THE PARKING AVAILABILITY USING Esp32:

**1.Real-time Parking Availability Display:**

The app should have a real-time display of parking availability data received from the Esp32. This can be shown on a map, where each parking spot is marked with a color indicating availability (e.g., green for available, red for occupied).

**2. Search and Filtering**:

Users should be able to search for parking availability near a specific location or filter the results based on criteria such as distance, price, or availability. This will help users find parking spots that suit their needs.

**3. Detailed Parking Information:**

Tapping on a parking spot on the map should provide more detailed information about that spot, such as pricing, restrictions, and any additional amenities (e.g., EV charging, disabled access).

**4. Navigation Integration:**

Users should be able to seamlessly integrate the parking availability data with navigation apps. This can be achieved by providing options to directly navigate to an available parking spot using popular navigation apps like Google Maps or Waze.

**5. Parking Spot Reservation:**

If supported, the app can also allow users to reserve a parking spot in advance. This can be facilitated by integrating with payment gateways and providing a secure and easy reservation process.

**6. User Reviews and Ratings:**

To ensure the accuracy of parking availability data, users should be able to provide reviews and ratings for parking spots they have used. This will help other users make informed decisions about which parking spots to choose.

**7. Notifications and Alerts:**

Users can set up notifications and alerts to receive updates on.