**SMART PARKING\_PHASE 4**

**1. Set Up Your Development Environment:**

**Install Flutter and Dart:** Follow the official Flutter documentation to set up your development environment for Flutter.

**Set up Raspberry Pi:** Configure your Raspberry Pi with the necessary sensors to detect parking availability. Establish a communication protocol, such as MQTT or WebSocket, to send data from Raspberry Pi to the Flutter app.

**2. Design the UI:**

* Design the app interface using Flutter widgets. Create a layout that displays parking availability information in real-time.
* Use Flutter components like List View, Card, and Text widgets to present the data elegantly.

**3. Implement Real-Time Data Communication:**

**Choose a communication protocol:** Use MQTT (Message Queuing Telemetry Transport) or WebSocket for real-time communication between the Raspberry Pi and the Flutter app.

**Implement communication in Flutter:** Use packages like mqtt\_client for MQTT or web\_socket\_channel for WebSocket to establish a connection between the app and the Raspberry Pi.

**4. Display Real-Time Parking Availability:**

**Receive data in Flutter:** Implement code to receive real-time parking availability data from the Raspberry Pi.

**Update UI dynamically:** Use set State method to update the UI in real-time as new parking availability data is received.

**Display parking availability:** Show the received data on the app interface, indicating which parking spots are available and which are occupied.

**5. Handle User Interactions:**

**Implement user interactions:** Allow users to interact with the app, such as selecting a specific parking area or viewing details of parking availability for different time slots.

**Implement filters and search:** Add options for users to filter parking availability based on location, time, or other criteria.

**6. Error Handling and Edge Cases:**

**Implement error handling:** Handle scenarios where the connection to the Raspberry Pi fails or when there's no data available.

**Handle edge cases:** Account for situations where the data received is inconsistent or invalid.

**7. Testing:**

**Test the app thoroughly:** Test the app on various devices and scenarios to ensure it functions correctly under different conditions.

**Debugging:** Use Flutter's debugging tools to identify and fix any issues in the code.

**8. Deployment:**

* Build the app for iOS and Android platforms.
* Deploy the Raspberry Pi server in the parking area.

**9. Maintenance and Updates:**

**Monitor app performance:** Keep an eye on user feedback and app analytics to identify areas for improvement.

**Regular updates:** Release updates with new features, bug fixes, and improvements based on user feedback.

**LINK FOR THE WEBSITE**

[**https://iotsmartparkingsystem1.mydurable.com/**](https://iotsmartparkingsystem1.mydurable.com/)