Real-Time Parking Availability Monitoring

This repository contains Python and Flutter code for a real-time parking availability monitoring system. The system comprises a Python server running on a Raspberry Pi and a Flutter app for mobile devices. The Python server simulates parking availability data and sends it to the Flutter app in real time via a WebSocket connection.

Python Server (Raspberry Pi)

Overview

The Python server on the Raspberry Pi acts as the data source for parking availability information. It uses the websockets library to establish WebSocket communication with connected clients, such as the Flutter app.

## \*\*Dependencies\*\*

- asyncio

- websockets

## Key Components

- `parking\_data`: An asynchronous function that simulates real-time parking availability data. In this example, it sends the message "Parking spaces available: 10" to connected clients every 5 seconds. You can replace this simulated data with an actual data source.

- `start\_server`: Sets up the WebSocket server to listen on IP "0.0.0.0" and port 8765.

- The server runs indefinitely, providing real-time data updates to connected clients.

## Dart Code for the Flutter App

## Overview

The Flutter app offers a user-friendly interface for monitoring the parking availability data received from the Raspberry Pi server. It establishes a WebSocket connection to receive and display real-time data updates.

# Key Components

- \*\*Dependencies\*\*:

- `package: flutter/ material`

- `package:web\_socket\_channel/io.dart`

- `MyApp`: The main application class that sets up the Flutter app.

- `ParkingAvailabilityScreen`: A stateful widget for displaying real-time parking availability data.

- The app initializes a WebSocket channel to connect to the Python server running on the Raspberry Pi. Make sure to replace `'your\_raspberry\_pi\_ip'` with the actual IP address.

- The app displays the current parking availability data, initially set to "Waiting for data...".

- It listens for incoming data from the WebSocket channel and updates the UI when new data is received.

- The app has an "initState" method to initialize the WebSocket connection and a "dispose" method to close the WebSocket when the app is closed.

- The UI is designed with an AppBar containing the app's title and a central area displaying the parking availability data with different font sizes for clarity.

## Getting Started

1. Set up your development environment for Flutter and Dart. Follow the official Flutter documentation [here](https://flutter.dev/docs/get-started/install).

2. Clone this repository to your development machine.

3. Run the Flutter app on your mobile device or emulator by executing `flutter run`.

4. Ensure that your Raspberry Pi server is running and accessible via the network. Update the IP address in the Flutter app to match your Raspberry Pi's IP.

5. Monitor real-time parking availability data through the Flutter app.

Feel free to adapt and expand on this code to create a complete parking availability monitoring system tailored to your needs.

PROGRAM:

A black background with a yellow and white flag

Description automatically generated with medium confidence

RESULT:

The provided Python code for the Raspberry Pi server and Dart code for the Flutter app create a real-time parking availability display system. Here's what the output and functionality look like:

\*\*Python Code (Raspberry Pi Server):\*\*

This Python code creates a WebSocket server that simulates real-time parking availability data. It sends a message with the number of available parking spaces (in this case, 10) every 5 seconds.

```python

A black and green screen with text

Description automatically generated

The WebSocket server listens for incoming connections on IP "0.0.0.0" and port 8765. When a client (the Flutter app) connects to it, it starts sending the availability data.

\*\*Dart Code (Flutter App):\*\*

The Flutter app, written in Dart, connects to the Raspberry Pi server via a WebSocket to receive real-time parking availability data.

Initially, it displays "Waiting for data..." on the screen. When the WebSocket connection is established, it updates the screen with the parking availability data received from the Raspberry Pi server.

The parking availability data is displayed in the app with two text widgets:

1. The first text widget displays "Parking Availability:" in a smaller font size.

2. The second text widget displays the actual parking availability data (e.g., "Parking spaces available: 10") in a larger font size with bold formatting.

The app updates this information in real-time as new data is received from the Raspberry Pi server.

Overall, this app provides a simple and user-friendly interface for displaying parking availability data received from the Raspberry Pi in real-time.

Please note that the app will display this information in an emulator or on a physical device running the Flutter app when it's correctly configured and connected to the Raspberry Pi server. You need to replace `'your\_raspberry\_pi\_ip'` with the actual IP address of your Raspberry Pi in the Dart code to establish the WebSocket connection.