Project: WhereBus - Bus Tracking for User, Driver, Admin

this project it will continue from esp32 project

Tech Stack:

- **Frontend:** Flutter
- **Backend:** PHP, MySQL
- Map API : flutter_map

Color Scheme:

- **Primary Color:** #40534C
- **Text (Active Button):** #FFFFFF
- **Cancel Button:** #7F7777
- **Menu (Active Button):** #677D6A
- **Send Location Button:** #40534C

Role-based Access:

- **User:** Can send location to Driver. (Marker removed after 20 minutes)
- **Driver:** Can monitor all user locations in real-time.

Authentication:

- **Register:** Username, Password
- **Login:** Username, Password

Navigation Menu display on (bottom screen)

- **Bus Icon:** Send and Watch Bus Location (disable for driver)
- **User Icon:** Can Logout from this page only Logout

Requirement

- All role
  - Replace Static name with actual name form API **on top left screen**
  - Fetch GPS location display on this screen
- **User side**
  - Add send Location function for user
- **Driver side**
  - Add send Location function for user

Next It will my esp32 code then my design

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

// WiFi credentials
const char* ssid = "unknow";
const char* password = "no-password-00";

// Server URL
const char* serverUrl = "http://192.168.1.96/api/gps.php";

// Bus ID
const char* bus_id = "1"; // Assign a unique bus ID

// GPS module setup
static const int RXPin = 16, TXPin = 17;
static const uint32_t GPSBaud = 9600;
TinyGPSPlus gps;
SoftwareSerial gpsSerial(RXPin, TXPin);

void setup() {
  Serial.begin(115200);
  gpsSerial.begin(GPSBaud);

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void loop() {
  // Check if GPS data is available
  while (gpsSerial.available() > 0) {
    gps.encode(gpsSerial.read());
    if (gps.location.isUpdated()) {
      double latitude = gps.location.lat();
      double longitude = gps.location.lng();

      Serial.print("Latitude: ");
      Serial.println(latitude, 6);
      Serial.print("Longitude: ");
      Serial.println(longitude, 6);
      String busStatus = "Online";
      // Send GPS data with bus_id to the server
      if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;
        http.begin(serverUrl);
        http.addHeader("Content-Type", "application/x-www-form-urlencoded");

        String httpRequestData = "bus_id=" + String(bus_id) + "&latitude=" + String(latitude, 6)
+ "&longitude=" + String(longitude, 6)+ "&status=" + busStatus;

        int httpResponseCode = http.POST(httpRequestData);
        if (httpResponseCode > 0) {
          Serial.println("Data sent successfully");
        } else {
          Serial.println("Error sending data");
        }

        http.end();
      } else {
        Serial.println("WiFi not connected");
      }

      // Wait 10 seconds before sending the next GPS data
      delay(10000);
    }
  }
}
```

gps.php (for store in database)

```
<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET, POST, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type");
// File: api/gps.php
include './config/database.php';

// Retrieve GPS data from POST request
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $bus_id = $_POST['bus_id'];
    $latitude = $_POST['latitude'];
    $longitude = $_POST['longitude'];

    // Validate the data
    if (is_numeric($bus_id) && is_numeric($latitude) && is_numeric($longitude)) {
        // Insert GPS data into the database
        $stmt = $pdo->prepare('INSERT INTO gps_data (bus_id, latitude, longitude,
timestamp) VALUES (?, ?, ?, NOW())');
        if ($stmt->execute([$bus_id, $latitude, $longitude])) {
            echo json_encode(['status' => 'success', 'message' => 'Data stored successfully']);
        } else {
            http_response_code(500);
            echo json_encode(['status' => 'error', 'message' => 'Failed to store data']);
        }
    } else {
        http_response_code(400);
        echo json_encode(['status' => 'error', 'message' => 'Invalid GPS data']);
    }
} else {
    http_response_code(405);
    echo json_encode(['status' => 'error', 'message' => 'Method not allowed']);
}
?>
```