

⚡ Charge Hub ⚡

Presented by :

George Ayman

Youssef Salem

Ismail Ayman

Thomas Nazih

Presented to : Eng/ Mohamed Shoshan

**Idea :**

After a great discussion with our teammates, we agreed that the coming technology is electric cars so we decided to help the users that need to charge their cars but we also decided to help the old people to make the app more viral and used by many users

**How we implement the code?**

First of the app we need to make a splash screen to make the app and the user satisfied

**A computer screen shot of a program

Description automatically generated**

A screen shot of a computer program

Description automatically generated

A screen shot of a phone

Description automatically generated

this is the code of the splash screen and the implementation

After that we need a login and sign-up page we used Firebase to authenticate and save the data and check it so,

**Authentication :**

**A screenshot of a phone

Description automatically generated**

**Login and Sign-up**

**A screenshot of a phone

Description automatically generated**

A screenshot of a computer

Description automatically generated

The implemented code :

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Sign-up code :**

import 'package:flutter/material.dart';  
import 'package:firebase\_auth/firebase\_auth.dart';  
import '../repeated/button.dart'; // Assuming this is a custom button widget  
import 'login.dart'; // Importing the login page  
  
class SignUp extends StatefulWidget {  
 @override  
 State<SignUp> createState() => \_SignUpState();  
}  
  
class \_SignUpState extends State<SignUp> {  
 final FirebaseAuth \_auth = FirebaseAuth.instance;  
  
 var \_usernameController = TextEditingController();  
 var \_emailController = TextEditingController();  
 var \_passwordController = TextEditingController();  
  
 @override  
 void dispose() {  
 \_usernameController.dispose();  
 \_emailController.dispose();  
 \_passwordController.dispose();  
 super.dispose();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: const Color.fromRGBO(31, 2, 75, 1.0), // Match login screen background  
 body: ListView(  
 padding: const EdgeInsets.symmetric(horizontal: 16.0),  
 children: [  
 const SizedBox(height: 20),  
 Padding(  
 padding: const EdgeInsets.all(10.0),  
 child: Container(  
 width: 300, // Set the same width and height as in login  
 height: 300,  
 child: FittedBox(  
 fit: BoxFit.cover, // Same BoxFit as login screen  
 child: Image.asset("assets/images/chargeHub.png"),  
 ),  
 ),  
 ),  
 const Center(  
 child: Text(  
 'Sign Up now For Free!',  
 style: TextStyle(  
 fontSize: 27,  
 color: Colors.white, // White text color  
 fontWeight: FontWeight.bold,  
 fontStyle: FontStyle.italic, // Italic to match login  
 ),  
 ),  
 ),  
 const SizedBox(height: 10),  
 const Center(  
 child: Text(  
 'Create new ChargeHub account',  
 style: TextStyle(  
 fontSize: 20,  
 color: Colors.white, // White text color  
 fontWeight: FontWeight.bold,  
 fontStyle: FontStyle.italic,  
 ),  
 ),  
 ),  
 const Center(  
 child: Text(  
 'and join the family!',  
 style: TextStyle(  
 fontSize: 20,  
 color: Colors.white, // White text color  
 fontWeight: FontWeight.bold,  
 fontStyle: FontStyle.italic,  
 ),  
 ),  
 ),  
 const SizedBox(height: 50),  
 // Username Input Field  
 Padding(  
 padding: const EdgeInsets.all(10.0),  
 child: TextFormField(  
 style: const TextStyle(color: Colors.black),  
 controller: \_usernameController,  
 decoration: InputDecoration(  
 filled: true,  
 fillColor: const Color(0xFFFFFFFF), // White field background  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(50),  
 borderSide: const BorderSide(color: Colors.black),  
 ),  
 labelText: 'Username',  
 hintText: 'Enter your name...',  
 labelStyle: const TextStyle(color: Colors.black),  
 ),  
 ),  
 ),  
 // Email Input Field  
 Padding(  
 padding: const EdgeInsets.all(10.0),  
 child: TextFormField(  
 style: const TextStyle(color: Colors.black),  
 controller: \_emailController,  
 decoration: InputDecoration(  
 filled: true,  
 fillColor: const Color(0xFFFFFFFF), // White field background  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(50),  
 borderSide: const BorderSide(color: Colors.black),  
 ),  
 labelText: 'Email',  
 labelStyle: const TextStyle(color: Colors.black),  
 hintText: 'Enter Email...',  
 ),  
 ),  
 ),  
 // Password Input Field  
 Padding(  
 padding: const EdgeInsets.all(10.0),  
 child: TextFormField(  
 style: const TextStyle(color: Colors.black),  
 obscureText: true,  
 controller: \_passwordController,  
 decoration: InputDecoration(  
 filled: true,  
 fillColor: const Color(0xFFFFFFFF), // White field background  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(50),  
 borderSide: const BorderSide(color: Colors.black),  
 ),  
 labelText: 'Password',  
 labelStyle: const TextStyle(color: Colors.black),  
 hintText: 'Enter Password...',  
 ),  
 ),  
 ),  
 const SizedBox(height: 20),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 16.0),  
 child: btnCal(  
 c: Colors.blue, // Blue button color to match login  
 text: "Sign Up",  
 event: \_signUp,  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.all(40.0),  
 child: Center(  
 child: InkWell(  
 onTap: () {  
 Navigator.of(context).pushReplacement(  
 MaterialPageRoute(  
 builder: (context) {  
 return Login(); // Navigate to the Login page  
 },  
 ),  
 );  
 },  
 child: RichText(  
 text: const TextSpan(  
 text: 'Do you have an existing account?',  
 style: TextStyle(  
 color: Colors.white, // White text color  
 fontSize: 16.0,  
 ),  
 children: <TextSpan>[  
 TextSpan(  
 text: ' Login',  
 style: TextStyle(  
 color: Colors.blue, // Blue text with underline  
 decoration: TextDecoration.underline,  
 ),  
 ),  
 ],  
 ),  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
  
 void \_signUp() async {  
 String email = \_emailController.text.trim();  
 String password = \_passwordController.text.trim();  
 String username = \_usernameController.text.trim();  
  
 if (email.isEmpty || password.isEmpty || username.isEmpty) {  
 ScaffoldMessenger.of(context).showSnackBar(  
 const SnackBar(content: Text('Please fill all fields')),  
 );  
 return;  
 }  
  
 try {  
 UserCredential userCredential =  
 await \_auth.createUserWithEmailAndPassword(  
 email: email,  
 password: password,  
 );  
  
 if (userCredential.user != null) {  
 ScaffoldMessenger.of(context).showSnackBar(  
 const SnackBar(content: Text('Sign Up Successful')),  
 );  
 // Navigate to the home page or perform other actions  
 }  
 } on FirebaseAuthException catch (e) {  
 ScaffoldMessenger.of(context).showSnackBar(  
 SnackBar(content: Text('Error: ${e.message}')),  
 );  
 }  
 }  
}

* After Signing in you are redirected to the Home page :

This page consists of 3 main

Buttons

1. A screenshot of a phone

   Description automatically generated**Charge hub:**

Which used to see the map

Add stations and know where

The are allocated

1. **Cars list:**

Which used to help the

Users if they want to buy a

New electric car

1. **Poll:**

Contain log-out and

The previous buttons

To easily navigate through

The app

Code for the home page :

import 'login.dart';  
import 'splachScreen.dart';  
import 'package:flutter/material.dart';  
import 'cars.dart'; // Make sure to import your CarGalleryPage  
import 'mapScreen.dart'; // Import MapScreen if not already imported  
  
class HomePage extends StatelessWidget {  
 final String username;  
  
 const HomePage({super.key, required this.username});  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: const Color.fromRGBO(31, 2, 75, 1),  
 appBar: AppBar(  
 backgroundColor: const Color.fromRGBO(31, 2, 75, 1.0),  
 title: Text('Welcome $username', style: const TextStyle(color: Colors.white)),  
 centerTitle: true,  
 leading: IconButton(  
 icon: const Icon(Icons.arrow\_back, color: Colors.white),  
 onPressed: () {  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) =>  
 Login(), // Navigate to CarGalleryPage  
 ),  
 );  
 },  
 ),  
 actions: [  
 PopupMenuButton<String>(  
 color: const Color.fromRGBO(31, 2, 75, 0.6),  
 icon: const Icon(Icons.menu, color: Colors.white),  
 onSelected: (value) {  
 switch (value) {  
 case 'Map':  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) => MapScreen(),  
 ),  
 );  
 break;  
 case 'Cars':  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) => CarGalleryPage(),  
 ),  
 );  
 break;  
 case 'Log out':  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) =>  
 SplashScreen(), // Replace with your actual SignupPage class  
 ),  
 );  
 break;  
 }  
 },  
 itemBuilder: (context) => [  
 const PopupMenuItem<String>(  
 value: 'Map',  
 child: Text('Map',  
 style: TextStyle(color: Colors.white)), // Black text  
 ),  
 const PopupMenuItem<String>(  
 value: 'Cars',  
 child: Text('Cars',  
 style: TextStyle(color: Colors.white)), // Black text  
 ),  
 const PopupMenuItem<String>(  
 value: 'Log out',  
 child: Text('Log out ',  
 style: TextStyle(color: Colors.white)), // Black text  
 ),  
 ],  
 ),  
 ],  
 ),  
 body: Column(  
 children: [  
 const SizedBox(height: 60),  
 GestureDetector(  
 onTap: () {  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) => MapScreen(), // Navigate to MapScreen  
 ),  
 );  
 },  
 child: Padding(  
 padding: const EdgeInsets.all(10.0),  
 child: Container(  
 width: double.infinity,  
 height: 250,  
 // Adjust height as needed  
 padding: const EdgeInsets.all(5),  
 decoration: BoxDecoration(  
 color: Colors.grey[850],  
 borderRadius: BorderRadius.circular(15),  
 ),  
 child: ClipRRect(  
 borderRadius: BorderRadius.circular(15),  
 child: Image.asset(  
 'assets/images/map.png',  
 // Path to your static map image  
 fit: BoxFit.cover,  
 ),  
 ),  
 ),  
 ),  
 ),  
 const SizedBox(height: 40), // Adjust spacing between images  
 GestureDetector(  
 onTap: () {  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) =>  
 CarGalleryPage(), // Navigate to CarGalleryPage  
 ),  
 );  
 },  
 child: Padding(  
 padding: const EdgeInsets.all(10.0),  
 child: Container(  
 width: double.infinity,  
 height: 400,  
 // Adjust height as needed  
 padding: const EdgeInsets.all(5),  
 decoration: BoxDecoration(  
 color: const Color.fromRGBO(51, 49, 48, 1),  
 borderRadius: BorderRadius.circular(15),  
 ),  
 child: ClipRRect(  
 borderRadius: BorderRadius.circular(15),  
 child: Image.asset(  
 'assets/images/carList.png',  
 // Replace with the path to your car image  
 fit: BoxFit.cover,  
 ),  
 ),  
 ),  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}

**A screenshot of a cell phone

Description automatically generatedA screenshot of a phone

Description automatically generatedcar list page:**

the data of this

cars are saved

inside a list and

can be updated

**code and implementing:**

import 'package:flutter/material.dart';  
  
import 'mapScreen.dart';  
import 'splachScreen.dart';  
  
class CarGalleryPage extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 backgroundColor: Color.fromRGBO(31, 2, 75, 1),  
 appBar: AppBar(  
  
 backgroundColor: Color.fromRGBO(31, 2, 75, 1),  
 title: Text('Car Collection', style: TextStyle(color: Colors.white)),  
 centerTitle: true,  
 actions: [  
 PopupMenuButton<String>(  
 color: const Color.fromRGBO(31, 2, 75, 0.6),  
 icon: const Icon(Icons.menu, color: Colors.white),  
 onSelected: (value) {  
 switch (value) {  
 case 'Map':  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) => MapScreen(),  
 ),  
 );  
 break;  
 case 'Cars':  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) => CarGalleryPage(),  
 ),  
 );  
 break;  
 case 'Log out':  
 Navigator.of(context).push(  
 MaterialPageRoute(  
 builder: (context) =>  
 SplashScreen(), // Replace with your actual SignupPage class  
 ),  
 );  
 break;  
 }  
 },  
 itemBuilder: (context) => [  
 const PopupMenuItem<String>(  
 value: 'Map',  
 child: Text('Map',  
 style: TextStyle(color: Colors.white)), // Black text  
 ),  
 const PopupMenuItem<String>(  
 value: 'Cars',  
 child: Text('Cars',  
 style: TextStyle(color: Colors.white)), // Black text  
 ),  
 const PopupMenuItem<String>(  
 value: 'Log out',  
 child: Text('Log out ',  
 style: TextStyle(color: Colors.white)), // Black text  
 ),  
 ],  
 ),  
 ],  
  
 ),  
 body: Padding(  
 padding: const EdgeInsets.all(16.0),  
 child: GridView.builder(  
 gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(  
 crossAxisCount: 2,  
 crossAxisSpacing: 16.0,  
 mainAxisSpacing: 16.0,  
 childAspectRatio: 1.5,  
 ),  
 itemCount: \_carImages.length,  
 itemBuilder: (context, index) {  
 final car = \_carImages[index];  
 return GestureDetector(  
 onTap: () {  
 \_showCarDetailsDialog(context, car);  
 },  
 child: ClipRRect(  
 borderRadius: BorderRadius.circular(15),  
 child: Stack(  
 fit: StackFit.expand,  
 children: [  
 Image.asset(  
 car['imagePath']!,  
 fit: BoxFit.cover,  
 ),  
 Positioned(  
 bottom: 0,  
 left: 0,  
 right: 0,  
 child: Container(  
 padding: EdgeInsets.all(8.0),  
 color: Colors.black54,  
 child: Text(  
 car['name']!,  
 style: TextStyle(  
 color: Colors.white,  
 fontWeight: FontWeight.bold,  
 fontSize: 16,  
 ),  
 overflow: TextOverflow.ellipsis,  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 },  
 ),  
 ),  
 );  
 }  
  
 final List<Map<String, String>> \_carImages = [  
 {'name': 'BMW i4', 'imagePath': 'assets/images/BMW i4.jpg'},  
 {'name': 'FORD Mustang', 'imagePath': 'assets/images/ford mustang.jpg'},  
 {'name': 'Tesla Model Y', 'imagePath': 'assets/images/Tesla model y.jpeg'},  
 {'name': 'Tesla Model 3', 'imagePath': 'assets/images/Tesla model3.jpeg'},  
 {'name': 'Tesla Model S', 'imagePath': 'assets/images/TESLA-MOTORS-Model-S-4693\_64.jpg'},  
 {'name': 'Tesla Model X', 'imagePath': 'assets/images/tesla model x.jpeg'},  
 ];  
  
 void \_showCarDetailsDialog(BuildContext context, Map<String, String> car) {  
 String details = '';  
 String quote = '';  
  
 switch (car['name']) {  
 case 'BMW i4':  
 details = 'The BMW i4 combines luxury and electric power, delivering a sporty and efficient ride. It offers a sleek design and advanced technology features.';  
 quote = '"The BMW i4 is the epitome of modern luxury and performance."';  
 break;  
 case 'FORD Mustang':  
 details = 'The FORD Mustang Mach-E is a high-performance electric SUV with classic Mustang styling. It provides a thrilling driving experience with cutting-edge tech.';  
 quote = '"The Mustang Mach-E offers an exhilarating drive while being environmentally conscious."';  
 break;  
 case 'Tesla Model Y':  
 details = 'The Tesla Model Y is a versatile electric crossover that combines efficiency with spaciousness. It features advanced autopilot capabilities and impressive range.';  
 quote = '"The Model Y brings Tesla’s innovation to a new level of versatility."';  
 break;  
 case 'Tesla Model 3':  
 details = 'The Tesla Model 3 is a popular electric sedan known for its range, performance, and affordability. It offers a minimalist interior and cutting-edge technology.';  
 quote = '"The Model 3 is a game-changer in the electric vehicle market."';  
 break;  
 case 'Tesla Model S':  
 details = 'The Tesla Model S is a luxury electric sedan with high performance and a long range. It boasts a sleek design, high-tech features, and impressive acceleration.';  
 quote = '"The Model S represents the pinnacle of electric vehicle engineering."';  
 break;  
 case 'Tesla Model X':  
 details = 'The Tesla Model X is an electric SUV known for its distinctive falcon-wing doors and advanced technology. It offers a spacious interior and exceptional range.';  
 quote = '"The Model X redefines what an electric SUV can be."';  
 break;  
 default:  
 details = 'No details available for this car.';  
 quote = 'No quote available.';  
 break;  
 }  
  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return AlertDialog(  
 backgroundColor: Color.fromRGBO(34, 37, 45, 1),  
 title: Text(  
 car['name']!,  
 style: TextStyle(color: Colors.white, fontSize: 20, fontWeight: FontWeight.bold),  
 ),  
 content: SingleChildScrollView(  
 child: Column(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 ClipRRect(  
 borderRadius: BorderRadius.circular(15),  
 child: Image.asset(  
 car['imagePath']!,  
 fit: BoxFit.cover,  
 height: 200,  
 ),  
 ),  
 SizedBox(height: 16),  
 Text(  
 details,  
 style: TextStyle(color: Colors.white70, fontSize: 16),  
 ),  
 SizedBox(height: 16),  
 Text(  
 quote,  
 style: TextStyle(color: Colors.tealAccent, fontSize: 14, fontStyle: FontStyle.italic),  
 ),  
 ],  
 ),  
 ),  
 actions: [  
 TextButton(  
 onPressed: () {  
 Navigator.of(context).pop();  
 },  
 child: Text('Close', style: TextStyle(color: Colors.tealAccent)),  
 ),  
 ],  
 );  
 },  
 );  
 }  
}

Map page:

In this page you can zoom in/out or add stations or to see the stations 1st screen for the whole screen 2nd screen for adding charging station you can add both charging or gas stations green for EV red for gas

A screen shot of a map

Description automatically generated

A screenshot of a phone

Description automatically generated

**Here is the code for this screen:**

import 'package:flutter/material.dart';  
import 'package:flutter\_bloc/flutter\_bloc.dart';  
import 'package:flutter\_map/flutter\_map.dart';  
import 'package:latlong2/latlong.dart';  
import 'package:url\_launcher/url\_launcher.dart';  
import '../bloc/logic.dart';  
  
class MapScreen extends StatefulWidget {  
 @override  
 \_MapScreenState createState() => \_MapScreenState();  
}  
  
class \_MapScreenState extends State<MapScreen> {  
 double \_zoomLevel = 13.0; // Initial zoom level for the map  
 LatLng \_mapCenter = LatLng(30.0444, 31.2357); // Initial map center (Cairo)  
 late MapController \_mapController; // For controlling the map programmatically  
  
 @override  
 void initState() {  
 super.initState();  
 \_mapController = MapController();  
 // Fetch stations when the screen is initialized  
 context.read<EVBloc>().add(FetchStationsEvent());  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: Text('Stations Map',style: TextStyle(color: Colors.white),),backgroundColor: Colors.deepPurple,  
 ),  
 body: BlocBuilder<EVBloc, EVState>(  
 builder: (context, state) {  
 if (state is EVLoading) {  
 return Center(child: CircularProgressIndicator());  
 } else if (state is EVStationsLoaded) {  
 final stations = state.stations;  
  
 return Stack(  
 children: [  
 // Flutter map  
 FlutterMap(  
 mapController: \_mapController,  
 options: MapOptions(  
 onTap: (tapPosition, latlng) {  
 \_addStationDialog(context, latlng);  
 },  
 ),  
 children: [  
 TileLayer(  
 urlTemplate:  
 'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',  
 subdomains: ['a', 'b', 'c'],  
 ),  
 MarkerLayer(  
 markers: stations.map((station) {  
 double lat =  
 station['x'] ?? 0.0; // Use 'x' for latitude  
 double lng =  
 station['y'] ?? 0.0; // Use 'y' for longitude  
 String stationName =  
 station['name'] ?? 'Unknown Station';  
 String stationType = station['type'] ?? 'Unknown';  
  
 // Refactored to use \_StationMarkerWidget  
 return Marker(  
 width: 100.0, // Adjusted size for better visibility  
 height: 100.0,  
 point: LatLng(lat, lng),  
 child: \_StationMarkerWidget(  
 icon: stationType == "EV"  
 ? Icons.ev\_station  
 : Icons.local\_gas\_station,  
 color:  
 stationType == "EV" ? Colors.green : Colors.red,  
 stationName: stationName,  
 latitude: lat,  
 longitude: lng,  
 onTap: () {  
 \_showStationDetailsDialog(  
 context, stationName, lat, lng, stationType);  
 },  
 ),  
 );  
 }).toList(),  
 ),  
 ],  
 ),  
 // Zoom in/out buttons at the bottom right corner  
 Positioned(  
 bottom: 80, // Place it above the add station button  
 right: 16, // Align to the right  
 child: Column(  
 children: [  
 FloatingActionButton(  
 heroTag: 'zoom\_in',  
 mini: true,  
 backgroundColor: Colors.deepPurple,  
 onPressed: () {  
 setState(() {  
 \_zoomLevel = (\_zoomLevel + 1).clamp(2.0, 18.0);  
 \_mapController.move(\_mapCenter, \_zoomLevel);  
 });  
 },  
 child: Icon(Icons.zoom\_in),  
 ),  
 SizedBox(height: 10),  
 FloatingActionButton(  
 heroTag: 'zoom\_out',  
 mini: true,  
 backgroundColor: Colors.deepPurple,  
 onPressed: () {  
 setState(() {  
 \_zoomLevel = (\_zoomLevel - 1).clamp(2.0, 18.0);  
 \_mapController.move(\_mapCenter, \_zoomLevel);  
 });  
 },  
 child: Icon(Icons.zoom\_out),  
 ),  
 ],  
 ),  
 ),  
 // Add Station button at the bottom right corner  
 Positioned(  
 bottom: 16,  
 right: 16,  
 child: FloatingActionButton(  
 heroTag: 'add\_station',  
 backgroundColor: Colors.blueAccent,  
 onPressed: () {  
 \_addStationDialog(context, \_mapCenter);  
 },  
 child: Icon(Icons.add),  
 ),  
 ),  
 ],  
 );  
 } else if (state is EVError) {  
 return Center(child: Text('Error: ${state.message}'));  
 }  
  
 return Center(child: Text('Unknown state'));  
 },  
 ),  
 );  
 }  
  
 void \_addStationDialog(BuildContext context, LatLng position) {  
 // Text editing controllers for additional fields  
 TextEditingController nameController = TextEditingController();  
 TextEditingController speedController = TextEditingController();  
  
 // Initialize dropdown selection for station type  
 String? selectedType;  
  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return AlertDialog(  
 backgroundColor: Colors.grey[900], // Dark background  
 title: Text(  
 'Add Charging Station',  
 style: TextStyle(color: Colors.white), // White text  
 ),  
 content: SingleChildScrollView(  
 child: Column(  
 mainAxisSize: MainAxisSize.min,  
 children: [  
 // Text fields for station details  
 TextField(  
 controller: nameController,  
 style: TextStyle(color: Colors.white), // White text input  
 decoration: InputDecoration(  
 labelText: 'Station Name',  
 labelStyle: TextStyle(color: Colors.grey), // Label color  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.white),  
 ),  
 focusedBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.blue),  
 ),  
 ),  
 ),  
 TextField(  
 controller: speedController,  
 style: TextStyle(color: Colors.white), // White text input  
 decoration: InputDecoration(  
 labelText: 'Charging Speed',  
 labelStyle: TextStyle(color: Colors.grey), // Label color  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.white),  
 ),  
 focusedBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Colors.blue),  
 ),  
 ),  
 ),  
 // Dropdown for station type  
 DropdownButtonFormField<String>(  
 dropdownColor: Colors.grey[800],  
 // Dark background for dropdown  
 decoration: InputDecoration(  
 labelText: 'Station Type',  
 labelStyle: TextStyle(color: Colors.grey), // Label color  
 ),  
 value: selectedType,  
 items: [  
 DropdownMenuItem(  
 value: "EV",  
 child: Text("EV charging",  
 style: TextStyle(color: Colors.white)),  
 ),  
 DropdownMenuItem(  
 value: "Gas",  
 child: Text("Gas station",  
 style: TextStyle(color: Colors.white)),  
 ),  
 ],  
 onChanged: (String? value) {  
 setState(() {  
 selectedType = value!;  
 });  
 },  
 validator: (value) =>  
 value == null ? 'Please select a type' : null,  
 ),  
 ],  
 ),  
 ),  
 actions: [  
 TextButton(  
 onPressed: () {  
 Navigator.of(context).pop();  
 },  
 child: Text('Cancel', style: TextStyle(color: Colors.white)),  
 ),  
 TextButton(  
 onPressed: () {  
 // Ensure station type is selected before proceeding  
 if (selectedType != null) {  
 // Dispatch AddStationEvent with position from the map  
 context.read<EVBloc>().add(AddStationEvent(  
 x: position.latitude,  
 y: position.longitude,  
 name: nameController.text,  
 speed: speedController.text,  
 type: selectedType!,  
 available: true, // Set available to true by default  
 ));  
 Navigator.of(context).pop();  
 }  
 },  
 child: Text('Add', style: TextStyle(color: Colors.blue)),  
 ),  
 ],  
 );  
 },  
 );  
 }  
  
 void \_confirmDeleteStation(BuildContext context, String stationId,  
 String stationName, double lat, double lng) {  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return AlertDialog(  
 title: Text('Station Details'),  
 content: Column(  
 mainAxisSize: MainAxisSize.min,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text('Name: $stationName'),  
 SizedBox(height: 8),  
 Text('Latitude: $lat'),  
 Text('Longitude: $lng'),  
 SizedBox(height: 16),  
 Text('Do you want to delete this station or visit?'),  
 ],  
 ),  
 actions: [  
 TextButton(  
 onPressed: () {  
 \_launchUrl(lat, lng); // Open Google Maps  
 },  
 child: Text('Visit'),  
 ),  
 TextButton(  
 onPressed: () {  
 context.read<EVBloc>().add(DeleteStationEvent(stationId));  
 Navigator.of(context).pop(); // Close the dialog  
 },  
 child: Text('Delete'),  
 ),  
 ],  
 );  
 },  
 );  
 }  
  
 // Function to launch Google Maps for a specific location  
 void \_launchUrl(double lat, double lng) async {  
 final String googleMapsUrl =  
 'https://www.google.com/maps/search/?api=1&query=$lat,$lng';  
  
 if (await canLaunch(googleMapsUrl)) {  
 await launch(googleMapsUrl);  
 } else {  
 throw 'Could not launch $googleMapsUrl';  
 }  
 }  
  
 // Add the missing method here  
 void \_showStationDetailsDialog(  
 BuildContext context, String name, double lat, double lng, String type) {  
 showDialog(  
 context: context,  
 builder: (BuildContext context) {  
 return AlertDialog(  
 backgroundColor: Colors.grey[900], // Dark background  
 title: Text(  
 name,  
 style: TextStyle(color: Colors.white), // White text for title  
 ),  
 content: Column(  
 mainAxisSize: MainAxisSize.min,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Text('Type: $type', style: TextStyle(color: Colors.white)),  
 // White text  
 Text('Latitude: $lat', style: TextStyle(color: Colors.white)),  
 // White text  
 Text('Longitude: $lng', style: TextStyle(color: Colors.white)),  
 // White text  
 ],  
 ),  
 actions: [  
 TextButton(  
 onPressed: () {  
 Navigator.of(context).pop();  
 },  
 child: Text('Close', style: TextStyle(color: Colors.white)),  
 ),  
 ],  
 );  
 },  
 );  
 }  
}  
  
// Custom widget for displaying station markers  
class \_StationMarkerWidget extends StatelessWidget {  
 final IconData icon;  
 final Color color;  
 final String stationName;  
 final double latitude;  
 final double longitude;  
 final VoidCallback onTap;  
  
 \_StationMarkerWidget({  
 required this.icon,  
 required this.color,  
 required this.stationName,  
 required this.latitude,  
 required this.longitude,  
 required this.onTap,  
 });  
  
 @override  
 Widget build(BuildContext context) {  
 return GestureDetector(  
 onTap: onTap, // Make the marker clickable  
 child: Column(  
 children: [  
 Icon(  
 icon,  
 color: color,  
 size: 50.0, // Adjusted size for better visibility  
 ),  
 SizedBox(height: 4),  
 Container(  
 padding: EdgeInsets.all(4),  
 decoration: BoxDecoration(  
 color: Colors.white,  
 borderRadius: BorderRadius.circular(4),  
 boxShadow: [  
 BoxShadow(  
 color: Colors.black26,  
 blurRadius: 2,  
 offset: Offset(0, 1),  
 ),  
 ],  
 ),  
 child: Text(  
 stationName,  
 style: TextStyle(  
 fontSize: 12,  
 fontWeight: FontWeight.bold,  
 ),  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}

**according to our instructor instructions we didn’t forget to make a state management for our project:**

* They are 2 files logic and state:

1st logic file :

import 'package:bloc/bloc.dart';  
import 'package:cloud\_firestore/cloud\_firestore.dart';  
import 'package:equatable/equatable.dart';  
import 'package:latlong2/latlong.dart'; // Import LatLng  
  
// Define the events  
abstract class Event extends Equatable {  
 @override  
 List<Object> get props => [];  
}  
  
class FetchStationsEvent extends Event {}  
  
class AddStationEvent extends Event {  
 final double x;  
 final double y;  
 final String name;  
 final String speed;  
 final bool available;  
 final String type;  
  
 AddStationEvent({  
 required this.x,  
 required this.y,  
 required this.name,  
 required this.speed,  
 required this.available,  
 required this.type,  
 });  
  
 @override  
 List<Object> get props => [x, y, name, speed, available, type];  
}  
  
class DeleteStationEvent extends Event {  
 final String stationId;  
  
 DeleteStationEvent(this.stationId);  
  
 @override  
 List<Object> get props => [stationId];  
}  
  
// Define the states  
abstract class EVState extends Equatable {  
 @override  
 List<Object> get props => [];  
}  
  
class EVLoading extends EVState {}  
  
class EVStationsLoaded extends EVState {  
 final List<Map<String, dynamic>> stations;  
  
 EVStationsLoaded(this.stations);  
  
 @override  
 List<Object> get props => [stations];  
}  
  
class EVError extends EVState {  
 final String message;  
  
 EVError(this.message);  
  
 @override  
 List<Object> get props => [message];  
}  
  
// Define the BLoC  
class EVBloc extends Bloc<Event, EVState> {  
 final FirebaseFirestore \_firestore = FirebaseFirestore.instance;  
  
 EVBloc() : super(EVLoading()) {  
 on<FetchStationsEvent>((event, emit) async {  
 await \_fetchStations(emit);  
 });  
  
 on<AddStationEvent>((event, emit) async {  
 await \_addStation(  
 event.x,  
 event.y,  
 event.name,  
 event.speed,  
 event.available,  
 event.type,  
 emit,  
 );  
 });  
  
 on<DeleteStationEvent>((event, emit) async {  
 await \_deleteStation(event.stationId, emit);  
 });  
 }  
  
 Future<void> \_fetchStations(Emitter<EVState> emit) async {  
 try {  
 emit(EVLoading());  
 QuerySnapshot snapshot = await \_firestore.collection('stations').get();  
 List<Map<String, dynamic>> stations = snapshot.docs.map((doc) {  
 final data = doc.data() as Map<String, dynamic>;  
 return {  
 'id': doc.id,  
 'x': data['x'] ?? 0.0, // Latitude  
 'y': data['y'] ?? 0.0, // Longitude  
 'name': data['name'] ?? 'Unknown Station',  
 'speed': data['speed'] ?? 'Unknown Speed',  
 'available': data['available'] ?? false,  
 'type': data['type'] ?? 'Unknown',  
 };  
 }).toList();  
 emit(EVStationsLoaded(stations));  
 } catch (e) {  
 emit(EVError('Error fetching stations: $e'));  
 }  
 }  
  
 Future<void> \_addStation(  
 double x,  
 double y,  
 String name,  
 String speed,  
 bool available,  
 String type,  
 Emitter<EVState> emit,  
 ) async {  
 try {  
 await \_firestore.collection('stations').add({  
 'x': x, // Latitude  
 'y': y, // Longitude  
 'name': name,  
 'speed': speed,  
 'available': available,  
 'type': type,  
 });  
  
 // Fetch updated stations  
 await \_fetchStations(emit);  
 } catch (e) {  
 emit(EVError('Error adding station: $e'));  
 }  
 }  
  
 Future<void> \_deleteStation(String stationId, Emitter<EVState> emit) async {  
 try {  
 await \_firestore.collection('stations').doc(stationId).delete();  
 // Fetch updated stations  
 await \_fetchStations(emit);  
 } catch (e) {  
 emit(EVError('Error deleting station: $e'));  
 }  
 }  
}

**2nd State File :**

// ev\_state.dart  
abstract class evState {}  
  
class evInitial extends evState {}  
  
class evLoading extends evState {}  
  
class evFetchSuccess extends evState {  
 final List<Map<String, dynamic>> stations;  
  
 evFetchSuccess(this.stations);  
}  
  
class evError extends evState {  
 final String error;  
  
 evError(this.error);  
}

**This was our project**

**Thanks**

