- As for the use of Flutter in the mobile app :

Flutter is a cross-platform framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase, so we use it in our Graduation project.

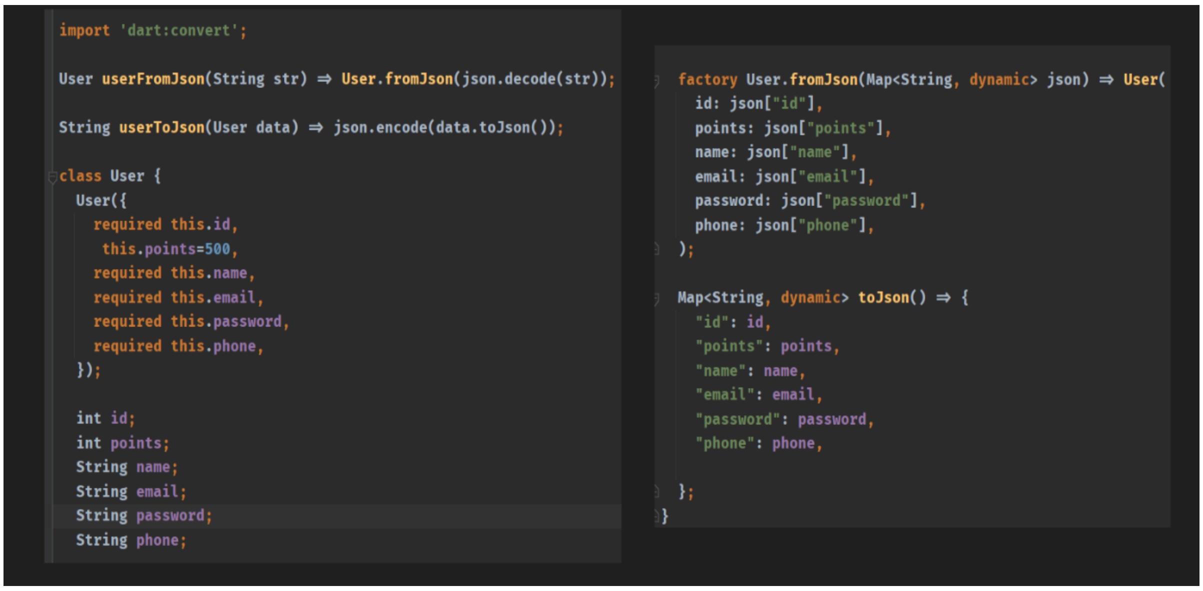
- Description of TrainDar Mobile:

We divide the project into:

- Models.
- Functions about API.
- Modules.
- Shared Components.
- Using Packages.

◆ First, We will talk about models:

Models describe our data in classes which have variables and converter functions fromJson() and toJson(), we divide them into three primary entities (Trains, Users, Stations) and have two sub-entities (Nearby Trains, and Close Stations).



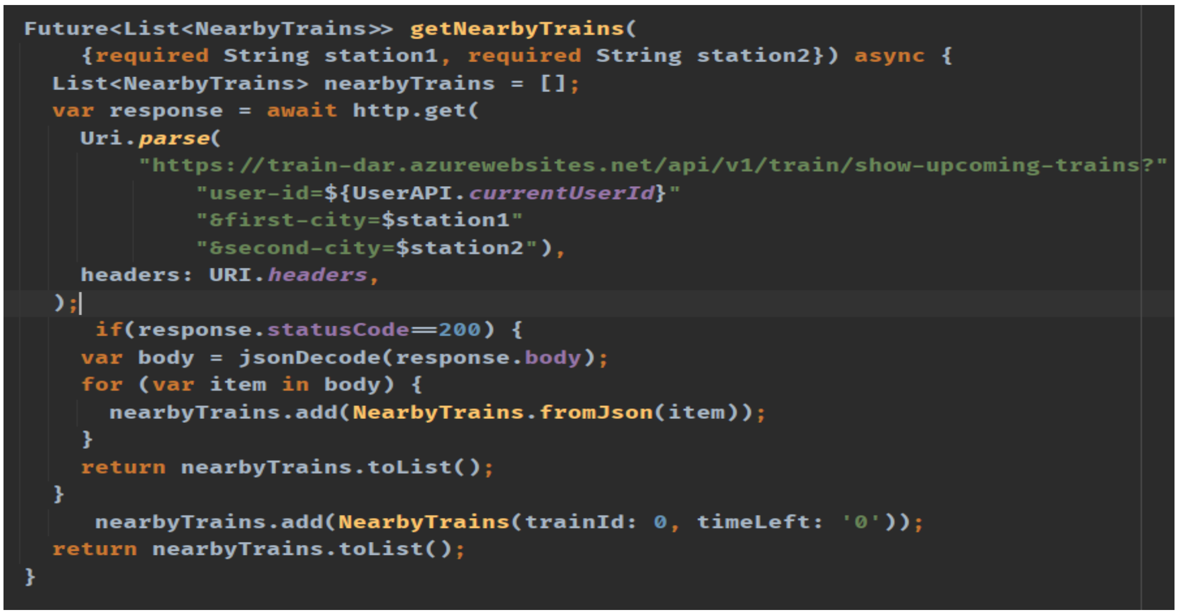
```dart
import 'dart:convert';

User userFromJson(String str) => User.fromJson(json.decode(str));

String userToJson(User data) => json.encode(data.toJson());

class User {
  User({
    required this.id,
     this.points=500,
    required this.name,
    required this.email,
    required this.password,
    required this.phone,
  });

  int id;
  int points;
  String name;
  String email;
  String password;
  String phone;
```

```dart
  factory User.fromJson(Map<String, dynamic> json) => User(
    id: json["id"],
    points: json["points"],
    name: json["name"],
    email: json["email"],
    password: json["password"],
    phone: json["phone"],
  );

  Map<String, dynamic> toJson() => {
    "id": id,
    "points": points,
    "name": name,
    "email": email,
    "password": password,
    "phone": phone,

  };
}
```

(User Model)

◆ Second, Functions of APIS:

We create a class that has future functions to handle with APIS. In these Future Functions, we use HTTP methods to do these operations (get, post, delete, update, put).
using async, await, and future because the data took more time to come from the back, after that we parse URL and Headers in the HTTP method which returns the response body, and we check if this response doesn’t have any exceptions, if not we navigate data from API to modules to display it, such as:

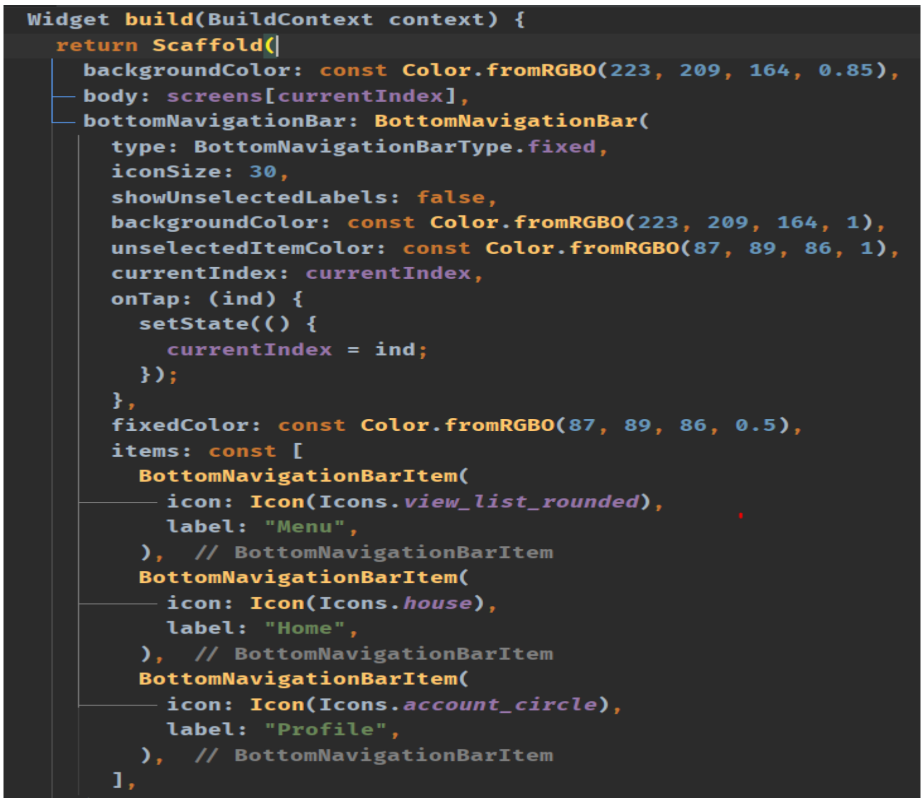


```dart
Future<List<NearbyTrains>> getNearbyTrains(
    {required String station1, required String station2}) async {
  List<NearbyTrains> nearbyTrains = [];
  var response = await http.get(
    Uri.parse(
        "https://train-dar.azurewebsites.net/api/v1/train/show-upcoming-trains?"
            "user-id=${UserAPI.currentUserId}"
            "&first-city=$station1"
            "&second-city=$station2"),
    headers: URI.headers,
  );
     if(response.statusCode==200) {
    var body = jsonDecode(response.body);
    for (var item in body) {
      nearbyTrains.add(NearbyTrains.fromJson(item));
    }
    return nearbyTrains.toList();
  }
     nearbyTrains.add(NearbyTrains(trainId: 0, timeLeft: '0'));
  return nearbyTrains.toList();
}
```

(Get Nearby Trains)

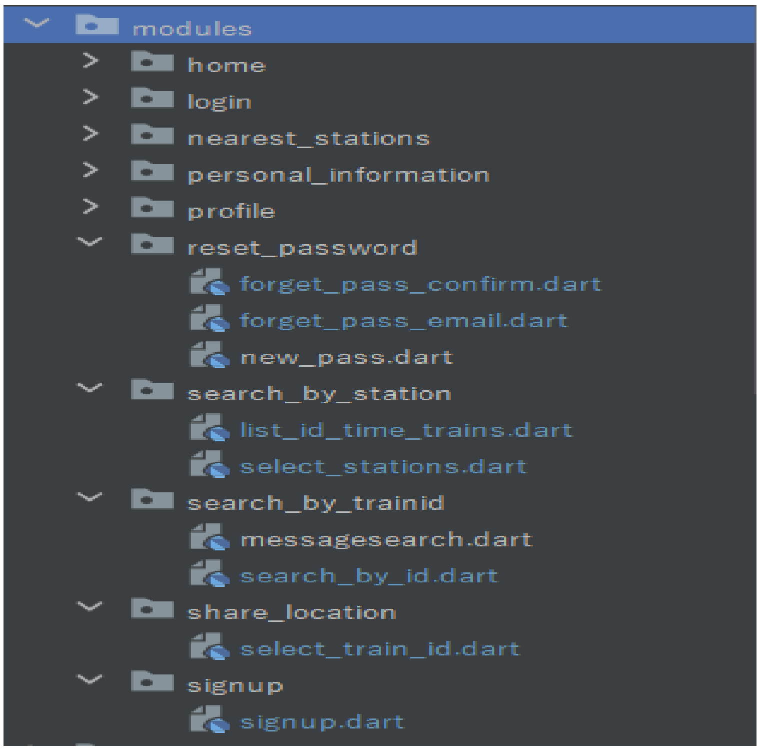
◆ third, Modules:

in this category, we display our design and data by creating classes extend from two main Widgets in Flutter(Stateful and Stateless), but at first, we create a layout by composing widgets to build more complex widgets and use initState(),setState() to handle the State Management.



```dart
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color.fromRGBO(223, 209, 164, 0.85),
    body: screens[currentIndex],
    bottomNavigationBar: BottomNavigationBar(
      type: BottomNavigationBarType.fixed,
      iconSize: 30,
      showUnselectedLabels: false,
      backgroundColor: const Color.fromRGBO(223, 209, 164, 1),
      unselectedItemColor: const Color.fromRGBO(87, 89, 86, 1),
      currentIndex: currentIndex,
      onTap: (ind) {
        setState(() {
          currentIndex = ind;
        });
      },
      fixedColor: const Color.fromRGBO(87, 89, 86, 0.5),
      items: const [
        BottomNavigationBarItem(
          icon: Icon(Icons.view_list_rounded),
          label: "Menu",
        ),  // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(Icons.house),
          label: "Home",
        ),  // BottomNavigationBarItem
        BottomNavigationBarItem(
          icon: Icon(Icons.account_circle),
          label: "Profile",
        ),  // BottomNavigationBarItem
      ],
```

(build function at home layout class)

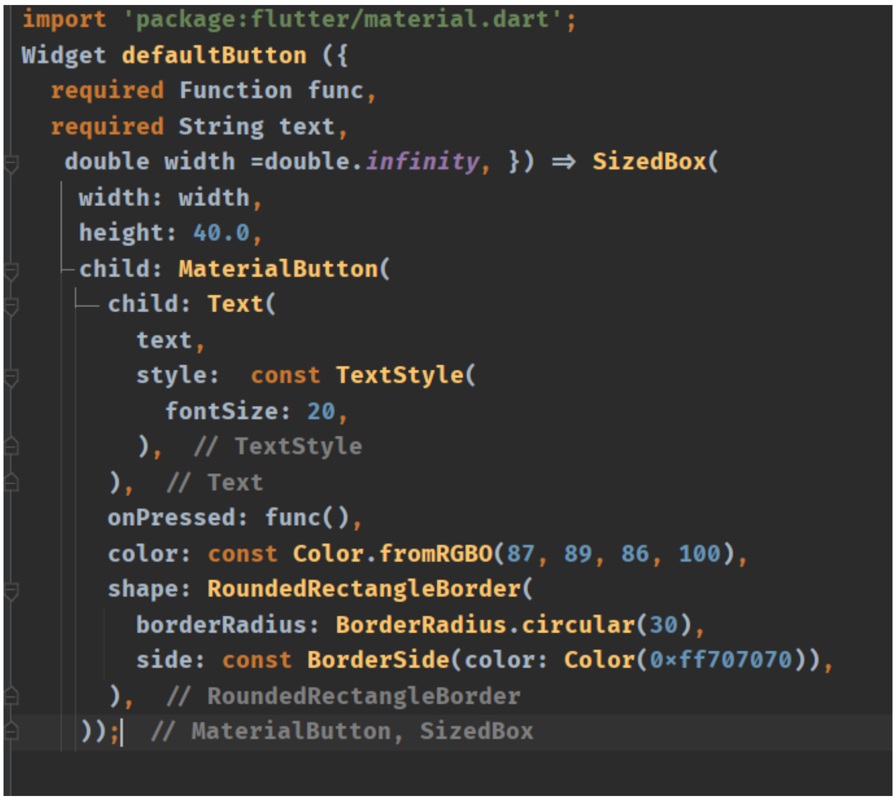


```
modules
  home
  login
  nearest_stations
  personal_information
  profile
  reset_password
    forget_pass_confirm.dart
    forget_pass_email.dart
    new_pass.dart
  search_by_station
    list_id_time_trains.dart
    select_stations.dart
  search_by_trainid
    messagesearch.dart
    search_by_id.dart
  share_location
    select_train_id.dart
  signup
    signup.dart
```

(our modules in the project)

◆ fourth, Shared Components :

There are some identical widgets, so we write one code in a separate class and then call it when needed it.



```
import 'package:flutter/material.dart';
Widget defaultButton ({
  required Function func,
  required String text,
   double width =double.infinity, }) ⇒ SizedBox(
    width: width,
    height: 40.0,
    child: MaterialButton(
      child: Text(
        text,
        style:  const TextStyle(
          fontSize: 20,
        ),  // TextStyle
      ),  // Text
      onPressed: func(),
      color: const Color.fromRGBO(87, 89, 86, 100),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(30),
        side: const BorderSide(color: Color(0×ff707070)),
      ),  // RoundedRectangleBorder
    ));  // MaterialButton, SizedBox
```

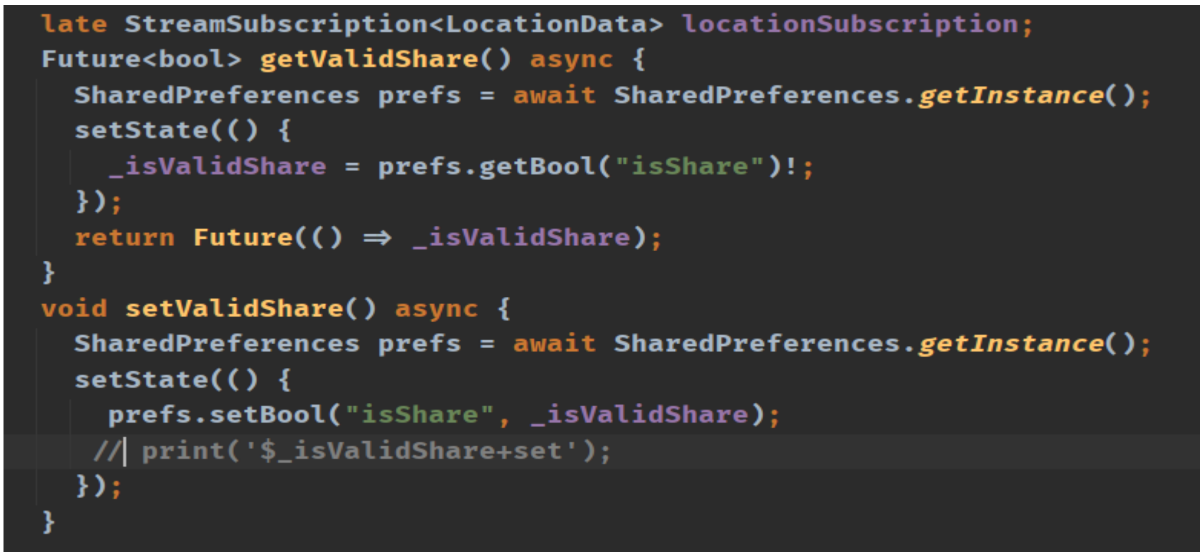
(default button in the UI)

◆ Fifth, Using Packages:

Packages:

1. **Shared Preference:**

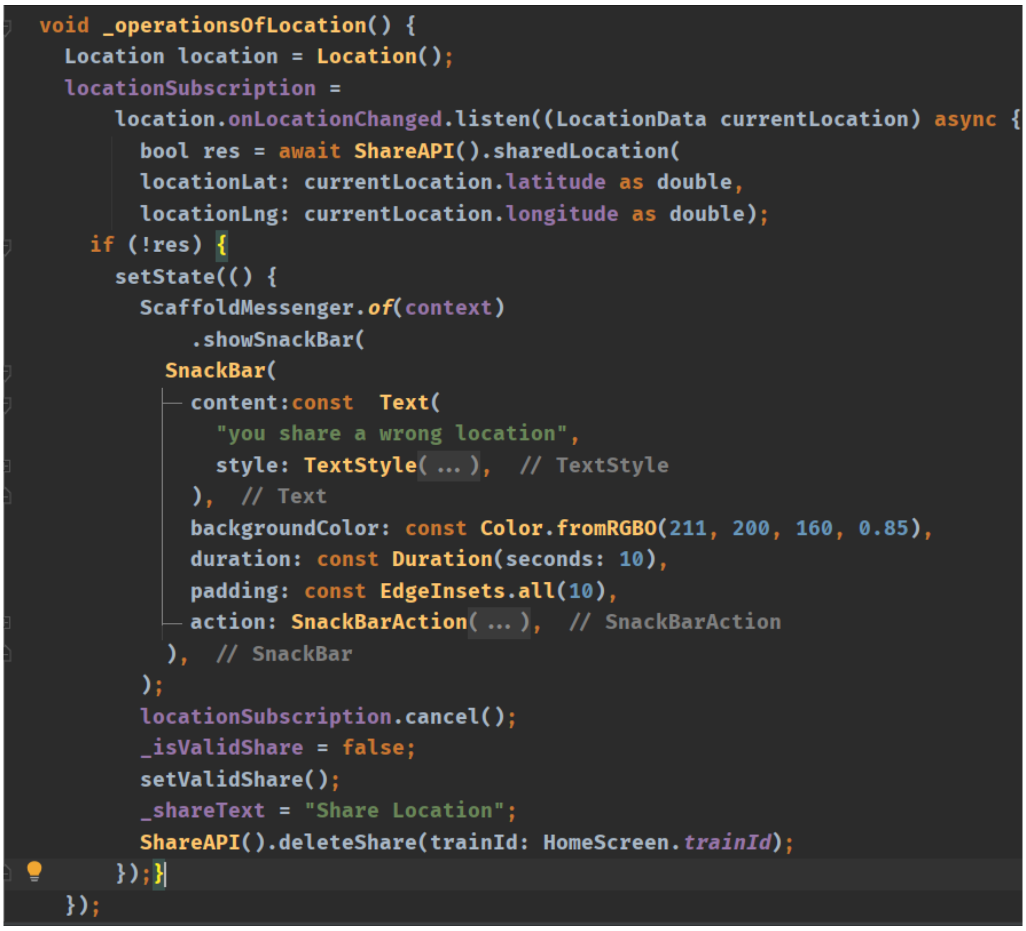
We use this package to store the condition of the user's location to check if this user shares his location or not.



```
late StreamSubscription<LocationData> locationSubscription;
Future<bool> getValidShare() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  setState(() {
    _isValidShare = prefs.getBool("isShare")!;
  });
  return Future(() ⇒ _isValidShare);
}
void setValidShare() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  setState(() {
    prefs.setBool("isShare", _isValidShare);
   //| print('$_isValidShare+set');
  });
}
```

2. **Location:**

This plugin for flutter handles getting a location on Android and iOS. It also provides callbacks when the location is changed and checks if the GPS is enabled or not.

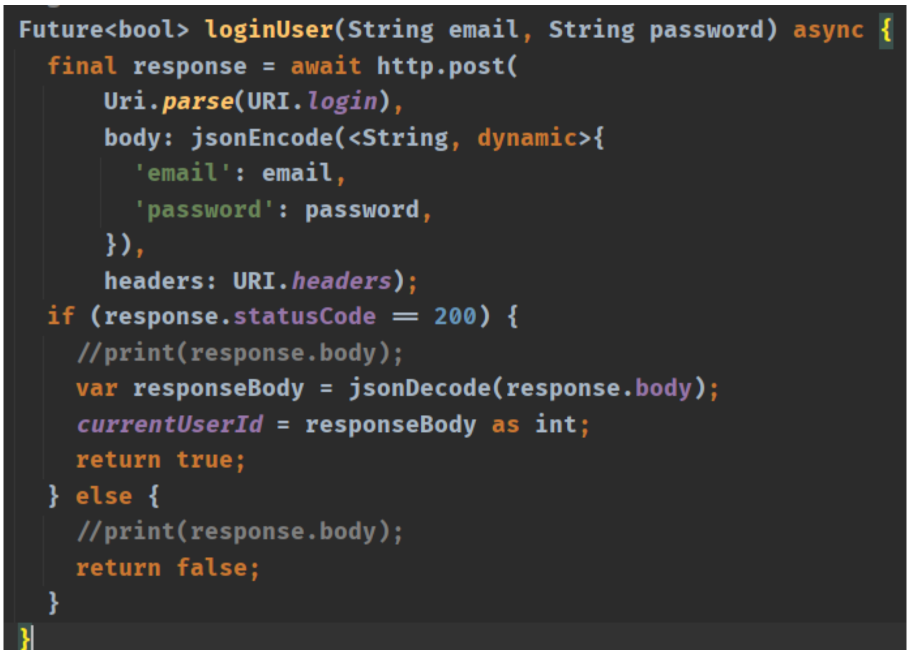


```
void _operationsOfLocation() {
  Location location = Location();
  locationSubscription =
      location.onLocationChanged.listen((LocationData currentLocation) async {
        bool res = await ShareAPI().sharedLocation(
        locationLat: currentLocation.latitude as double,
        locationLng: currentLocation.longitude as double);
    if (!res) {
      setState(() {
        ScaffoldMessenger.of(context)
            .showSnackBar(
          SnackBar(
            content:const  Text(
              "you share a wrong location",
              style: TextStyle( ... ),  // TextStyle
            ),  // Text
            backgroundColor: const Color.fromRGBO(211, 200, 160, 0.85),
            duration: const Duration(seconds: 10),
            padding: const EdgeInsets.all(10),
            action: SnackBarAction( ... ),  // SnackBarAction
          ),  // SnackBar
        );
        locationSubscription.cancel();
        _isValidShare = false;
        setValidShare();
        _shareText = "Share Location";
        ShareAPI().deleteShare(trainId: HomeScreen.trainId);
      });}
  });
```

(on changed Location function)

3. **HTTP:**

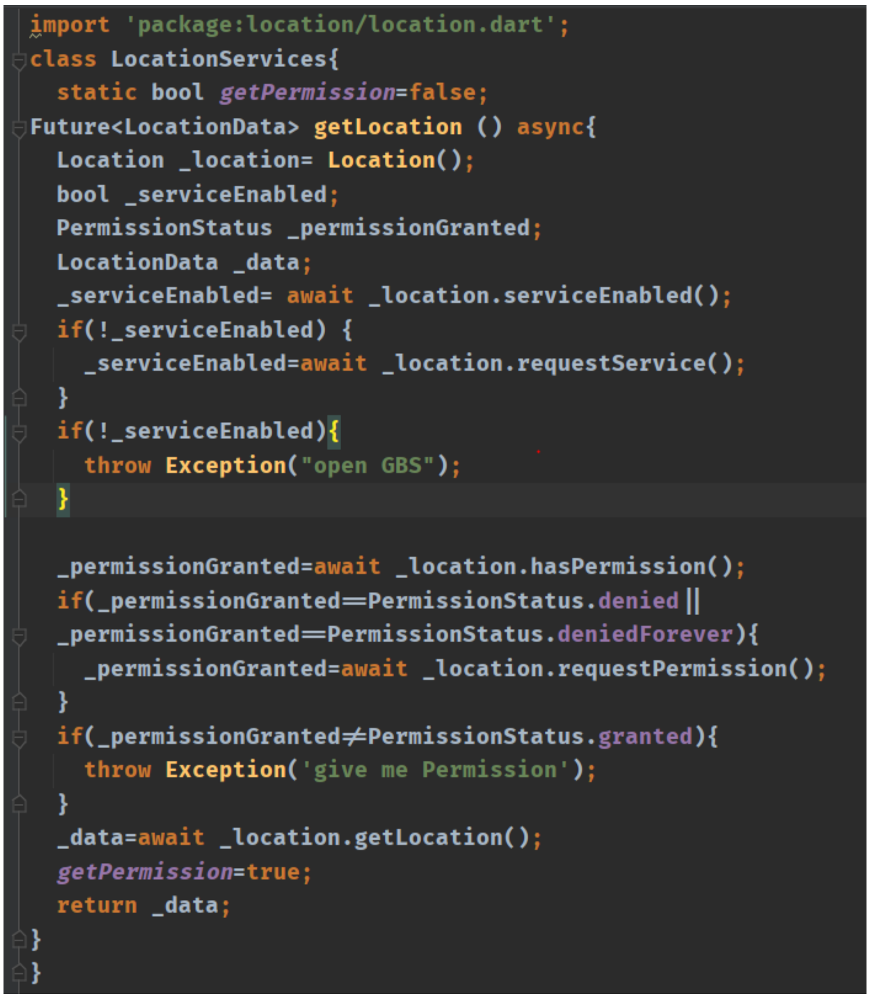
This package contains a set of high-level functions and classes that make it easy to consume HTTP resources. We use it to deal with APIS.



```
Future<bool> loginUser(String email, String password) async {
  final response = await http.post(
      Uri.parse(URI.login),
      body: jsonEncode(<String, dynamic>{
        'email': email,
        'password': password,
      }),
      headers: URI.headers);
  if (response.statusCode == 200) {
    //print(response.body);
    var responseBody = jsonDecode(response.body);
    currentUserId = responseBody as int;
    return true;
  } else {
    //print(response.body);
    return false;
  }
}
```

4. **Google_Maps_Flutter:**

A Flutter plugin that provides a Google Maps widget from google cloud.
we create a project on the cloud and enable google map services to get the key and add it in the permission access to flutter code.



```dart
import 'package:location/location.dart';
class LocationServices{
  static bool getPermission=false;
Future<LocationData> getLocation () async{
  Location _location= Location();
  bool _serviceEnabled;
  PermissionStatus _permissionGranted;
  LocationData _data;
  _serviceEnabled= await _location.serviceEnabled();
  if(!_serviceEnabled) {
    _serviceEnabled=await _location.requestService();
  }
  if(!_serviceEnabled){
    throw Exception("open GBS");
  }

  _permissionGranted=await _location.hasPermission();
  if(_permissionGranted==PermissionStatus.denied ||
  _permissionGranted==PermissionStatus.deniedForever){
    _permissionGranted=await _location.requestPermission();
  }
  if(_permissionGranted!=PermissionStatus.granted){
    throw Exception('give me Permission');
  }
  _data=await _location.getLocation();
  getPermission=true;
  return _data;
}
}
```

(test permissions).