

Flutter & Dart Complete Guide

Prepared by Eng. Yousef Salah Kassem

DAFL 3
Integration

Lecture 10



Chapter
10

Table Of Content

- Firebase Social Provider Authentication
- Firebase Notifications
- Google Maps



Fake Store API Flutter Application

Project

Problem Story: The FakeStore API Application

You have been hired as a **backend developer** for a new startup called **FakeStore CLI Services**. The company's mission is to create a command-line tool that interacts with the [FakeStore API](#) to enable users to manage their shopping experience directly from the terminal.

Your task is to create a **Flutter application** that provides users with the following features:



Project

1. Login:

Users should be able to log in using their username and password to authenticate with the FakeStore API. Upon successful login, they receive a token, which will be displayed.

2. View Product Categories:

Users can fetch and view a list of product categories available in the store.

3. View Products by Category:

Users can choose a specific category and view all products under that category. Each product should display its:

- a. ID
- b. Title
- c. Price



Project

User-Friendly Experience:

The application should use a **menu-based system** to guide users through the options.

Error Handling:

The application must handle errors gracefully. For example:

- Invalid credentials during login.
- Network issues or timeouts.
- Invalid inputs, such as choosing a non-existent category.

Code Requirements:

- Follow **OOP principles** by dividing responsibilities into classes like `ApiClient`, `AuthService`, and `ProductService`.
- Use the **Dio** package for HTTP requests.
- Ensure the application is modular, readable, and maintainable.



Project

User-Friendly Experience:

The application should use a **menu-based system** to guide users through the options.

Error Handling:

The application must handle errors gracefully. For example:

- Invalid credentials during login.
- Network issues or timeouts.
- Invalid inputs, such as choosing a non-existent category.

Code Requirements:

- Follow **OOP principles** by dividing responsibilities into classes like `ApiClient`, `AuthService`, and `ProductService`.
- Use the **Dio** package for HTTP requests.
- Ensure the application is modular, readable, and maintainable.



Project

- Add the **dio** package to your **pubspec.yaml**:

```
dependencies:  
  dio: ^5.0.0
```



User login

```
fetch('https://fakestoreapi.com/auth/login',{
  method:'POST',
  body:JSON.stringify({
    username: "mor_2314",
    password: "83r5^_"
  })
})
.then(res=>res.json())
.then(json=>console.log(json))
```

Hide output

```
//output
{
  token: "eyJhbGciOiJIUzI1NiIsInR"
```

Products

Get all products

```
fetch('https://fakestoreapi.com/products')  
  .then(res=>res.json())  
  .then(json=>console.log(json))
```

Hide output

```
//output  
[  
  {  
    id:1,  
    title:'...',  
    price:'...',  
    category:'...',  
    description:'...',  
    image:'...'  
  },  
  /*...*/  
  {  
    id:30,  
    title:'...',  
    price:'...',  
    category:'...',  
    description:'...',  
    image:'...'  
  }  
]
```

Get all categories

```
fetch('https://fakestoreapi.com/products/categories')  
  .then(res=>res.json())  
  .then(json=>console.log(json))
```

Hide output

```
//output  
[  
  "electronics",  
  "jewelery",  
  "men's clothing",  
  "women's clothing"  
]
```

<https://www.figma.com/design/jDCfm30VHot486hrCqrCsX/Grocery-App--Big-Cart---Community-?node-id=2464-1073&p=f&t=tZJLKJnXI7erYLxH-0>

Splash1-4

Get Discounts On All Products

Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy



Get started

Splash1-3

Buy Premium Quality Fruits

Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy



Get started

Splash1-2



Splash1-1



Auth1-welcome

A mobile app interface for a grocery store. The top half features a large image of a woman smiling while pushing a shopping cart filled with various fruits and vegetables in a grocery store aisle. Below the image, the word "Welcome" is displayed in a bold, sans-serif font. Underneath, there is a line of placeholder text: "Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy". At the bottom, there is a white button with the Google logo and the text "Continue with google". Below this button, there is a link that says "Already have an account ? login".

9:41

Welcome

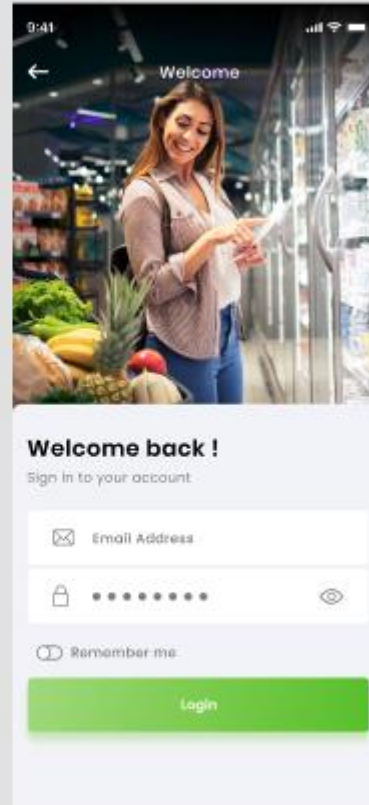
Welcome

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy

 Continue with google

Already have an account ? [login](#)

Auth1-login

A mobile app interface for a grocery store, showing the login screen. The top half features a large image of a woman smiling while looking at a smartphone in a grocery store aisle. Below the image, the text "Welcome back !" is displayed in a bold, sans-serif font. Underneath, there is a line of placeholder text: "Sign in to your account". Below this, there are two input fields: the first is labeled "Email Address" and the second is for a password, represented by a series of dots and a toggle icon. Below the password field, there is a checkbox labeled "Remember me". At the bottom, there is a large green button with the text "login".

9:41

Welcome

Welcome back !

Sign in to your account

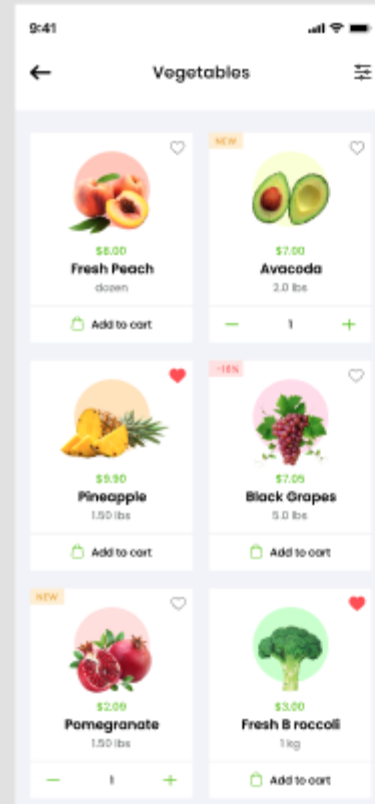
 Email Address

 •••••••• 

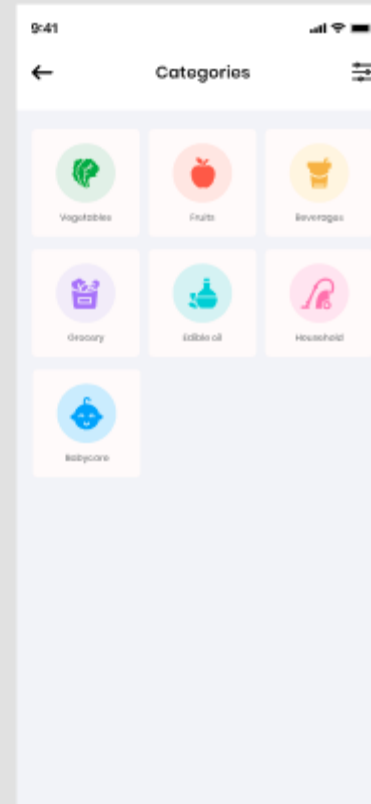
☐ Remember me

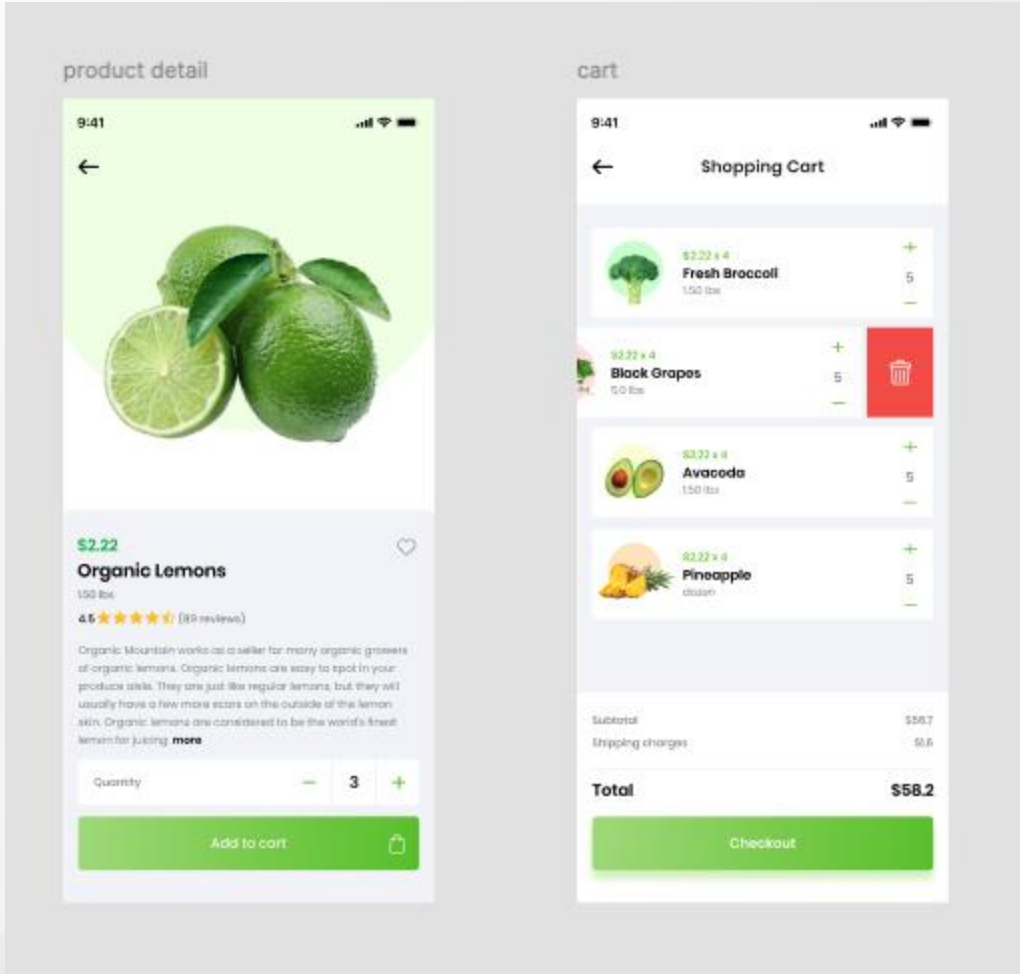
login

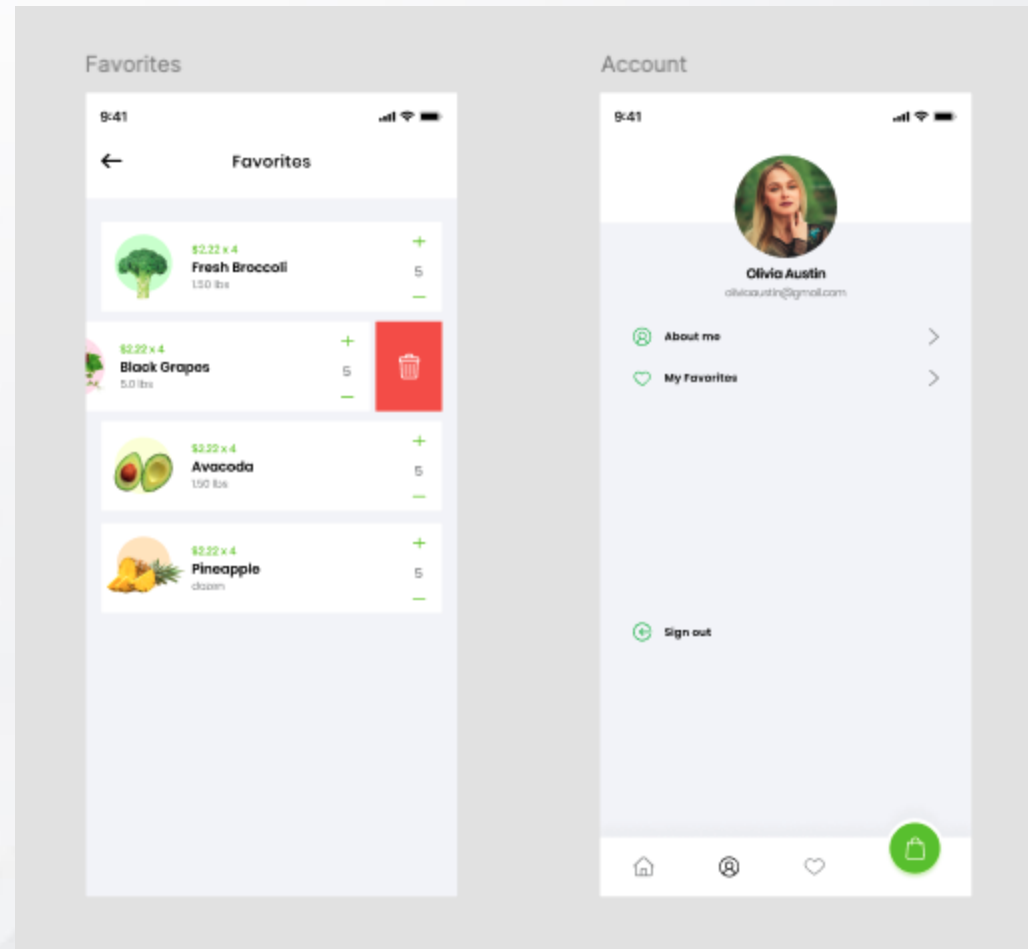
Prodcuts

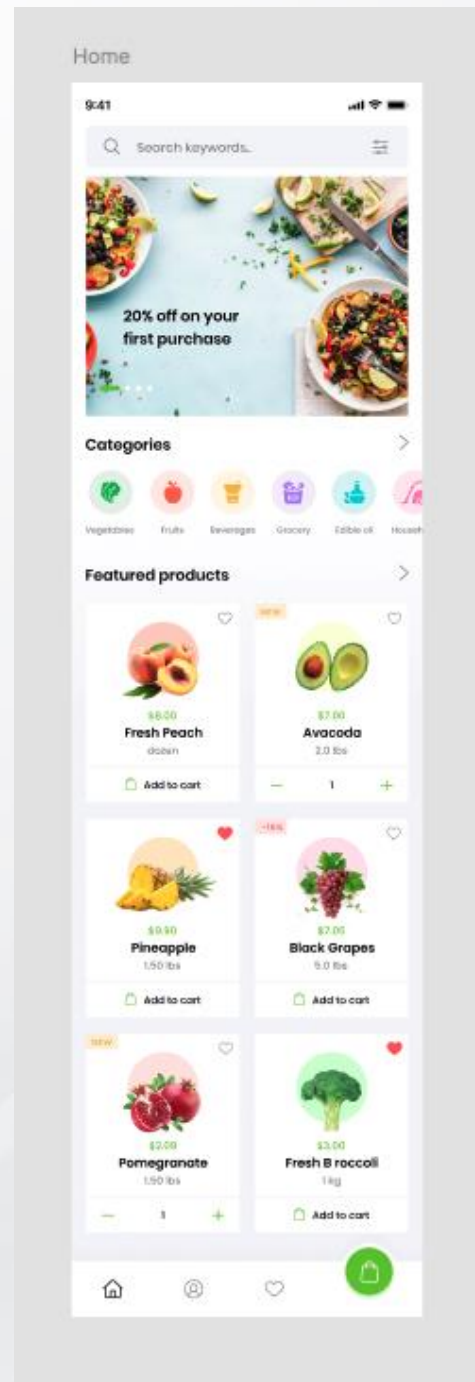


Category









About me

9:41   

← About me

Personal Details

 Russell Austin

 russell.partner@gmail.com

 +1 202 555 0142

Change Password

 Current password

 ••••• 

 Confirm password

Save settings

GET

https://fakestoreapi.com/products

Send

Params

Authorization

Headers (7)

Body

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (14)

Test Results

200 OK

302 ms

4.95 KB

Save Response

...

{ } JSON

Preview

Visualize

1

[

2

{

3

"id": 1,

4

"title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",

5

"price": 109.95,

6

"description": "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",

7

"category": "men's clothing",

8

"image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",

9

"rating": {

10

"rate": 3.9,

11

"count": 120

12

}

13

},

14

{

15

"id": 2,

16

"title": "Mens Casual Premium Slim Fit T-Shirts ",

17

"price": 22.3,

18

"description": "Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing. And Solid stitched shirts with round neck made for durability and a great fit for casual fashion wear and diehard baseball fans. The Henley style round neckline includes a three-button placket.",

19

"category": "men's clothing",

20

"image": "https://fakestoreapi.com/img/71-3HjGNDUL._AC_SY879._SX._UX._SY._UY_.jpg",

21

"rating": {

22

"rate": 4.1,

23

"count": 259

24

}

25

}

GET

https://fakestoreapi.com/products/1

Send

Params

Authorization

Headers (7)

Body

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	⋮ Bulk Edit
	Key	Value	Description	

Body

Cookies

Headers (14)

Test Results

↺

200 OK

• 275 ms • 959 B •

🌐

📄 Save Response

⋮

{{ JSON

Preview

Visualize

⋮

⌵

⌵

🔍

📄

🔗

```

1  {
2    "id": 1,
3    "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
4    "price": 109.95,
5    "description": "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",
6    "category": "men's clothing",
7    "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",
8    "rating": {
9      "rate": 3.9,
10     "count": 120
11   }
12 }
```




fake api / New Request

DELETE



https://fakestoreapi.com/users/1

Params

Authorization

Headers (7)

Body

Scripts

Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Login Authentication

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:google_sign_in/google_sign_in.dart';
import 'package:midocean_app/core/error/failure.dart';

final firebaseAuthProvider = Provider((ref) => FirebaseAuthService());

class FirebaseAuthService {
  final _firebaseAuth = FirebaseAuth.instance;
  final _googleSignIn = GoogleSignIn(scopes: ['email', 'profile']);

  User? get user => _firebaseAuth.currentUser;

  Stream<User?> get authStateChanges => _firebaseAuth.authStateChanges();

  Future<Either<Failure, UserCredential>> signInWithGoogle() async {
    try {
      final googleUser = await _googleSignIn.signIn();
      if (googleUser == null) {
        return const Left(ServerFailure(message: 'Google sign in failed'));
      }
      final googleAuth = await googleUser.authentication;
      final credential = GoogleAuthProvider.credential(
        accessToken: googleAuth.accessToken,
        idToken: googleAuth.idToken,
      );
      return Right(await _firebaseAuth.signInWithCredential(credential));
    } on FirebaseAuthException catch (e) {
      return Left(ServerFailure(message: e.message ?? e.toString()));
    }
  }

  Future<void> signOut() async {
    await _firebaseAuth.signOut();
    await _googleSignIn.signOut();
  }
}
```

```
final authControllerProvider = ChangeNotifierProvider((ref) {  
  return AuthController(  
    ref.watch(baseCachingServiceProvider(CachingType.secure)),  
    ref.watch(firebaseAuthServiceServiceProvider),  
  ); // AuthController  
}); // ChangeNotifierProvider  
  
class AuthController extends ChangeNotifier {  
  AuthController(this._cachingService, this._firebaseAuthService);  
  
  final BaseCachingService _cachingService;  
  final FirebaseAuthService _firebaseAuthService;  
  
  bool _isLoggedIn = false;  
  
  bool get isLoggedIn => _isLoggedIn;  
  
  Future<void> checkIfUserLoggedIn() async {  
    _isLoggedIn = await _cachingService.readString('token') != null;  
    debugPrint('Is logged in: $_isLoggedIn');  
    notifyListeners();  
  }  
  
  Future<void> logout() async {  
    await _firebaseAuthService.signOut();  
    await _cachingService.delete('token');  
    _isLoggedIn = false;  
    notifyListeners();  
  }  
}
```

```
Future<void> googleLogin() async {  
    state = const AsyncValue.loading();  
    final result =  
    await ref.read(firebaseAuthServiceProvider).signInWithGoogle();  
    result.fold(  
        (failure) {  
            state = AsyncValue.error(  
                failure.message,  
                failure.stackTrace ?? StackTrace.current,  
            );  
        },  
        (data) {  
            final user = data.user;  
            if (user != null) {  
                final token = user.uid;  
                ref  
                    .read(baseCachingServiceProvider(CachingType.secure))  
                    .writeString('token', token);  
                state = AsyncValue.data(token);  
            }  
        },  
    );  
}
```

Firestore Notifications

```
class FirebaseNotificationService {
  static final FirebaseMessaging _firebaseMessaging =
    FirebaseMessaging.instance;
  static final FlutterLocalNotificationsPlugin
    _flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();

  static Future<void> initialize() async {
    FirebaseMessaging.onMessage.listen((message) {
      final notification = message.notification;
      final android = message.notification?.android;

      if (kDebugMode) {
        print('Title data: ${notification?.title}');
        print('Description data: ${notification?.body}');
        print('Message android: ${android?.count}');
      }

      if (Platform.isAndroid) {
        initLocalNotification(message);
        showNotification(message);
      }

      if (Platform.isIOS) {
        setupForegroundMessage();
      }
    });
  }
}
```

```
static Future<String> getDeviceToken() async {
  if (kDebugMode) {
    print('Device token: ${await _firebaseMessaging.getToken()}');
  }
  return await _firebaseMessaging.getToken() ?? '';
}

static Future<void> initLocalNotification(RemoteMessage message) async {
  const initializationSettingsAndroid = AndroidInitializationSettings(
    '@mipmap/ic_launcher',
  );

  const initializationSettingsIOS = DarwinInitializationSettings();

  const initializationSettings = InitializationSettings(
    android: initializationSettingsAndroid,
    iOS: initializationSettingsIOS,
  );

  await _flutterLocalNotificationsPlugin.initialize(
    initializationSettings,
    onDidReceiveNotificationResponse: (payload) {
      print(payload.payload);
    },
    onDidReceiveBackgroundNotificationResponse: (payload) {
      print(payload.payload);
    },
  );
}
```

```
static Future<void> requestNotificationPermission() async {  
    await _firebaseMessaging.requestPermission();  
}  
  
static Future<void> setupInteractMessage() async {  
    final initialMessage = await FirebaseMessaging.instance.getInitialMessage();  
    if (initialMessage != null) {  
        print(initialMessage.data);  
    }  
  
    FirebaseMessaging.onMessageOpenedApp.listen((message) {  
        print(message.data);  
    });  
}
```

```
static Future<void> showNotification(RemoteMessage message) async {  
  const android = AndroidNotificationDetails(  
    'channel_id',  
    'channel_name',  
    importance: Importance.max,  
    priority: Priority.high,  
    icon: 'mipmap/ic_launcher',  
    ticker: 'test',  
  );  
  
  const ios = DarwinNotificationDetails(  
    presentAlert: true,  
    presentBadge: true,  
    presentSound: true,  
  );  
  
  const platform = NotificationDetails(android: android, iOS: ios);  
  
  await _flutterLocalNotificationsPlugin.show(  
    0,  
    message.notification?.title,  
    message.notification?.body,  
    platform,  
  );  
}  
  
static Future<void> setupForegroundMessage() async {  
  FirebaseMessaging.instance.setForegroundNotificationPresentationOptions(  
    alert: true,  
    badge: true,  
    sound: true,  
  );  
}
```

Google Maps

```
class MapsScreen extends StatefulWidget {  
  const MapsScreen({super.key});  
  
  @override  
  State<MapsScreen> createState() => _MapsScreenState();  
}  
  
class _MapsScreenState extends State<MapsScreen> {  
  final testPosition = const LatLng(37.42796133580664, -122.085749655962);  
  
  late CameraPosition initialCameraPosition;  
  
  final Completer<GoogleMapController> _controller =  
    Completer<GoogleMapController>();  
  
  @override  
  void initState() {  
    initialCameraPosition = CameraPosition(target: testPosition, zoom: 14.4746);  
  
    super.initState();  
  }  
  
  Future<void> _goToCurrentPosition() async {  
    final GoogleMapController controller = await _controller.future;  
    final position = await Geolocator.getCurrentPosition();  
    final currentPosition = LatLng(position.latitude, position.longitude);  
    controller.animateCamera(  
      CameraUpdate.newCameraPosition(  
        CameraPosition(target: currentPosition, zoom: 14.4746),  
      ),  
    );  
  }  
}
```

```
@override
Widget build(BuildContext context) {
  return Stack(
    children: [
      GoogleMap(
        onMapCreated: _controller.complete,
        initialCameraPosition: initialCameraPosition,
        mapType: MapType.hybrid,
        markers: {
          Marker(
            markerId: const MarkerId('currentPosition'),
            position: testPosition,
          ), // Marker
        },
        myLocationEnabled: true,
        trafficEnabled: true,
        indoorViewEnabled: true,
      ), // GoogleMap
      Positioned(
        left: 0,
        bottom: 0,
        child: GestureDetector(
          onTap: () async {
            await _goToCurrentPosition();
          },
          child: const CircleAvatar(
            radius: 20,
            backgroundColor: Colors.white,
            child: Icon(Icons.location_on),
          ), // CircleAvatar
        ), // GestureDetector
      ), // Positioned
    ],
  ); // Stack
}
```

What is Next ?



Thank You