

Nama : Akmal Muhamad Firdaus

NIM : 1301204188

Jurnal Praktikum Modul 11

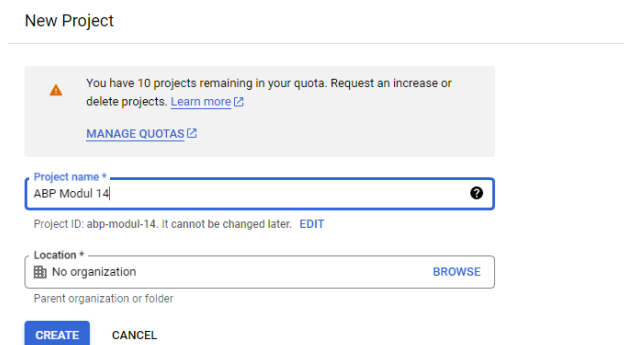
1. Google Maps API

Google Maps API adalah antarmuka pemrograman aplikasi (API) yang disediakan oleh Google untuk mengintegrasikan dan menggunakan fitur-fitur dari Google Maps dalam aplikasi atau situs web. API ini memungkinkan pengembang perangkat lunak untuk mengakses berbagai data dan fungsi dari Google Maps, seperti peta interaktif, informasi lokasi, rute, pencarian tempat, dan banyak lagi.

1.1. Cara mendapatkan API Key Google Maps

Cara implementasi Google API pada flutter dapat dilakukan dengan menggunakan packages Google Maps. Tahapan dalam menambahkan Google Maps API dapat mengikuti langkah-langkah berikut :

1. Buka [Google Cloud Console](#) dan buat project baru.



New Project

⚠ You have 10 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *
ABP Modul 14

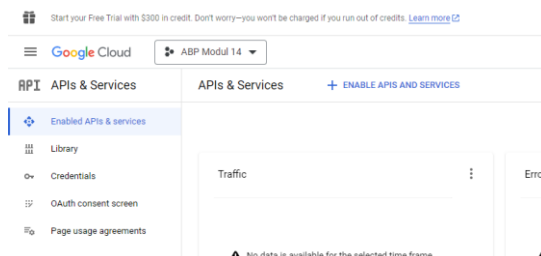
Project ID: abp-modul-14. It cannot be changed later. [EDIT](#)

Location *
No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

2. Buka tab Enable API dan klik Enable API and Services



Start your Free Trial with \$300 in credit. Don't worry—you won't be charged if you run out of credits. [Learn more](#)

Google Cloud ABP Modul 14

APIs & Services [+ ENABLE APIS AND SERVICES](#)

Enabled APIs & services

Library

Credentials

OAuth consent screen

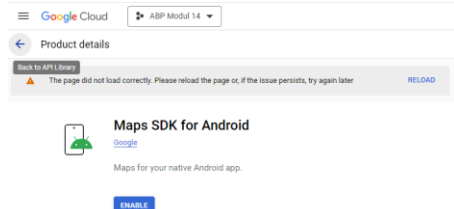
Page usage agreements

Traffic

Error

⚠ No data is available for the selected time frame.

3. Klik "Maps SDK for Android" dan tekan enable



Google Cloud ABP Modul 14

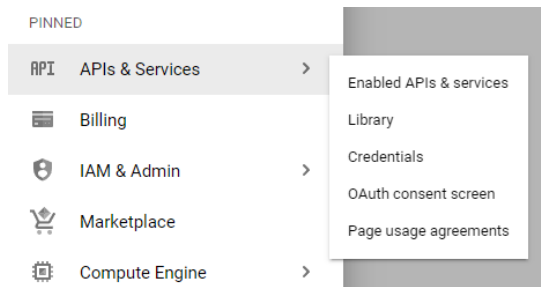
Product details

Maps SDK for Android

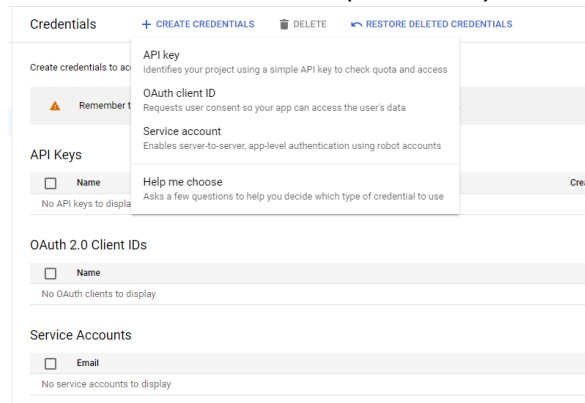
Maps for your native Android app.

[ENABLE](#)

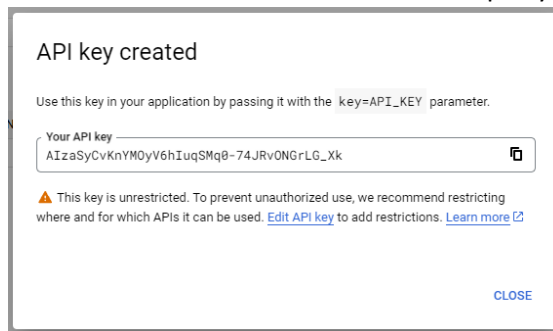
4. Untuk mendapatkan API Key, buka tab API lalu klik Credentials



5. Klik create credential dan pilih API Key



6. Setelah itu maka akan muncul modal apikey dan bisa langsung kita gunakan apikey tersebut.



1.2. Implementasi Google Maps Pada Flutter

1. Set minSdkVersion di android/app/build.gradle menjadi seperti berikut:

```
defaultConfig {
    applicationId "com.example.modul14"
    minSdkVersion 20
    targetSdkVersion flutter.targetSdkVersion
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
}
```

2. Tambahkan API key dan permission menggunakan GPS pada manifest aplikasi

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.modul14">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <application
        android:label="modul14"
        android:name="${applicationName}"
        android:icon="@mipmap/ic_launcher">
        <meta-data android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyCvKnYMOyV6hIuqSMq8-74JRvONGrLG_Xk"/>
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|locale|layoutDirection|fontScale|screenLayout|density|uiMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <meta-data
                android:name="io.flutter.embedding.android.NormalTheme"
                android:resource="@style/NormalTheme"
            />
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
</manifest>
```

1.3. Menambahkan Packages dan Menggunakan Google Maps

1. Buka pub.dev dan cari google_maps_flutter, lalu copy versi depedensi terbaru
2. Masukan kedalam pubspec.yaml

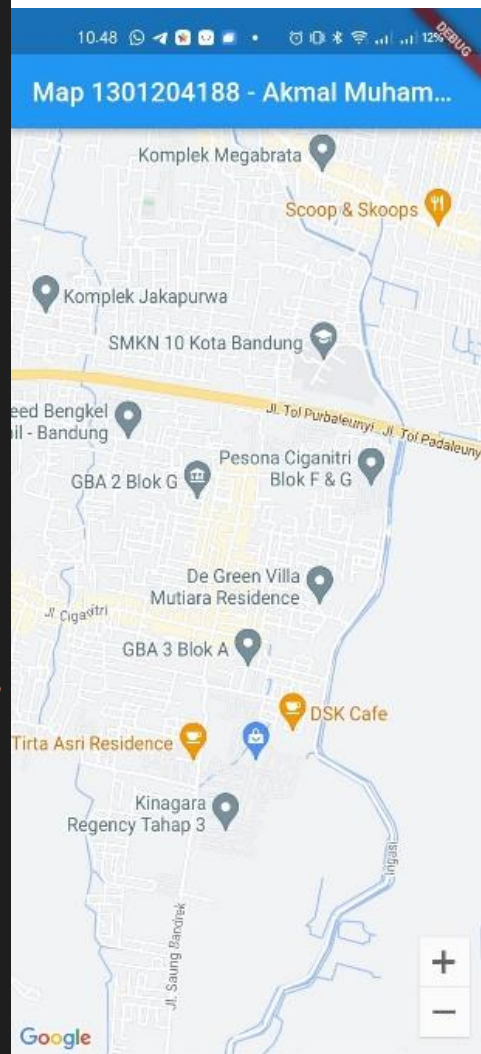
```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.2  
  google_maps_flutter: ^2.2.8
```

3. Import packages ke dalam file Dart

```
import 'package:google_maps_flutter/google_maps_flutter.dart';
```

4. Masukan full codingan berikut untuk menggunakan widget Maps

```
import 'package:flutter/material.dart';  
import 'package:google_maps_flutter/google_maps_flutter.dart';  
  
Run | Debug | Profile  
void main() => runApp(MyApp());  
class MyApp extends StatelessWidget {  
  const MyApp({super.key});  
  @override  
  Widget build(BuildContext context) {  
    return const MaterialApp(  
      home: MyMapScreen(),  
    ); // MaterialApp  
  }  
}  
class MyMapScreen extends StatefulWidget {  
  const MyMapScreen({super.key});  
  
  @override  
  // ignore: library_private_types_in_public_api  
  _MyMapScreenState createState() => _MyMapScreenState();  
}  
class _MyMapScreenState extends State<MyMapScreen> {  
  GoogleMapController? mapController;  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: const Text('Map 1301204188 - Akmal Muhamad Firdaus'),  
      ), // AppBar  
      body: GoogleMap(  
        onMapCreated: (controller) {  
          setState(() {  
            mapController = controller;  
          });  
        },  
        initialCameraPosition: const CameraPosition(  
          target: LatLng(-6.970439057048023, 107.65600083402336),  
          zoom: 15.0,  
        ), // CameraPosition  
      ), // GoogleMap  
    ); // Scaffold  
  }  
}
```



1.4. Menambahkan Akses Lokasi Kita Pada Manifest

Untuk dapat menampilkan posisi pengguna, sistem memerlukan permission location dari OS yang digunakan, dalam kasus ini saya menggunakan Android.

1. Tambahkan permission_handler untuk menghandle permintaan (request) segala permission pada pubspec.yaml

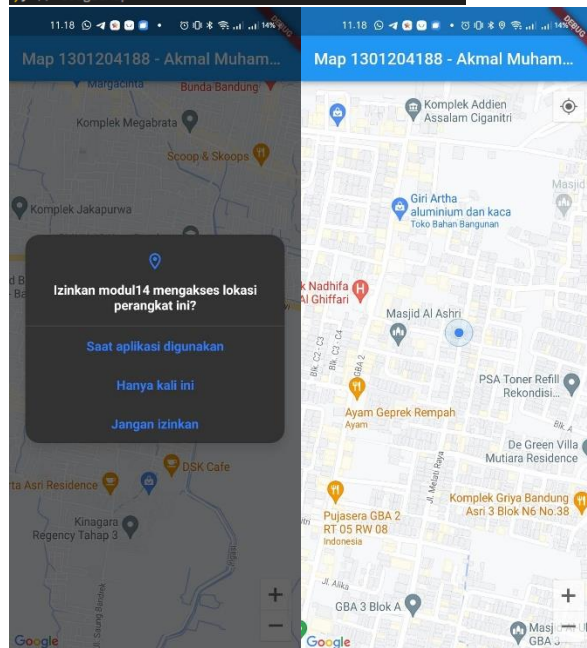
```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.2  
  google_maps_flutter: ^2.2.8  
  permission_handler: ^10.2.0
```

2. Tambahkan codingan berikut untuk meminta permission yang dibutuhkan (location)

```
@override  
void initState() {  
  super.initState();  
  _requestGpsPermission();  
}  
  
Future<void> _requestGpsPermission() async {  
  final status = await Permission.locationWhenInUse.request();  
  if (status.isGranted) {  
    // Permission granted  
  } else if (status.isDenied) {  
    // Permission denied  
  } else if (status.isPermanentlyDenied) {  
    // Permission permanently denied  
    openAppSettings();  
  }  
}
```

3. Setelah menambahkan function handler permission, cukup tambahkan myLocationEnabled dengan value true pada GoogleMap dan secara otomatis ketika pengguna membuka aplikasi akan meminta permission untuk mengakses lokasi.

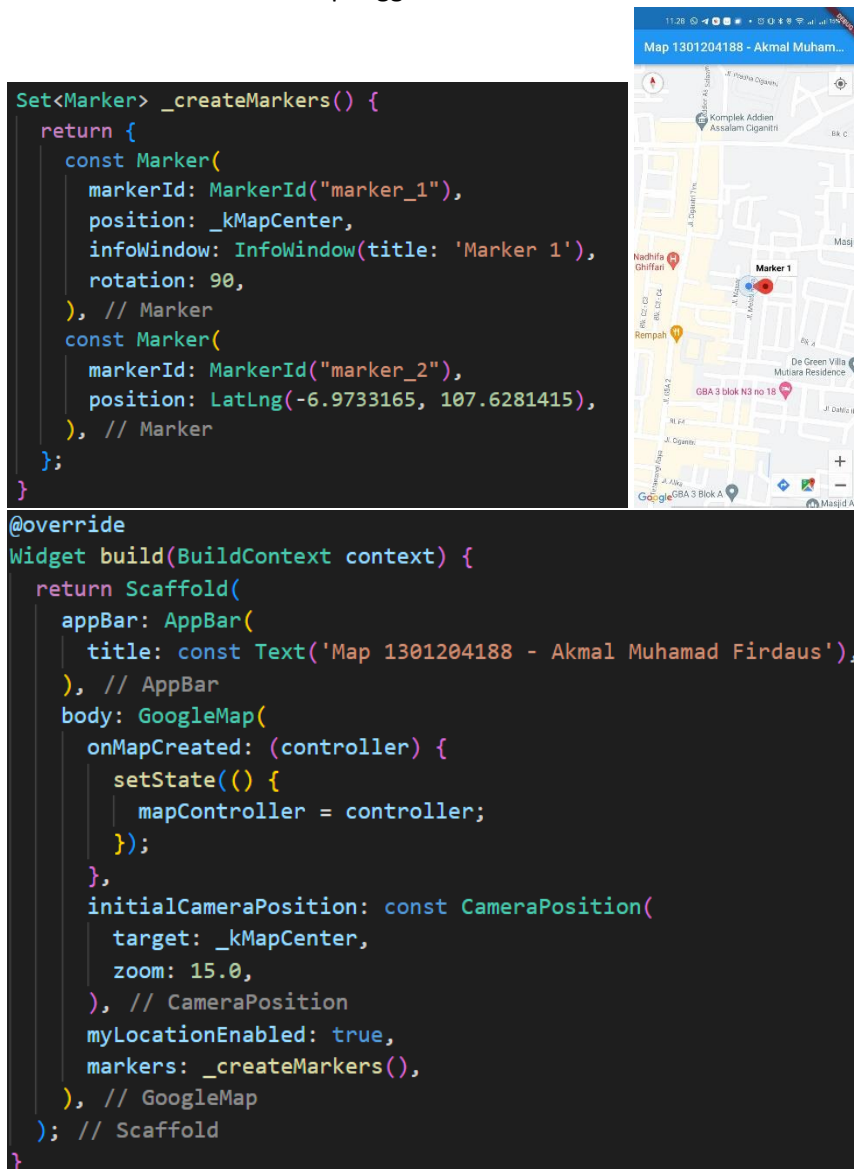
```
body: GoogleMap(  
  onMapCreated: (controller) {  
    setState(() {  
      mapController = controller;  
    });  
  },  
  initialCameraPosition: const CameraPosition(  
    target: LatLng(-6.970439057048023, 107.65600883402336),  
    zoom: 15.0,  
  ), // CameraPosition  
  myLocationEnabled: true,  
), // GoogleMap
```



1.5. Menambahkan Marker pada Google Maps

Marker pada Google Maps adalah objek yang digunakan untuk menandai atau menunjukkan lokasi spesifik di peta. Marker biasanya ditampilkan sebagai ikon atau tanda yang dapat ditempatkan di koordinat tertentu pada peta. Marker dapat memiliki berbagai properti seperti ID, posisi (koordinat), judul, deskripsi, ikon khusus, dan lainnya.

Berikut adalah cara penggunaan marker:



2. Place Picker

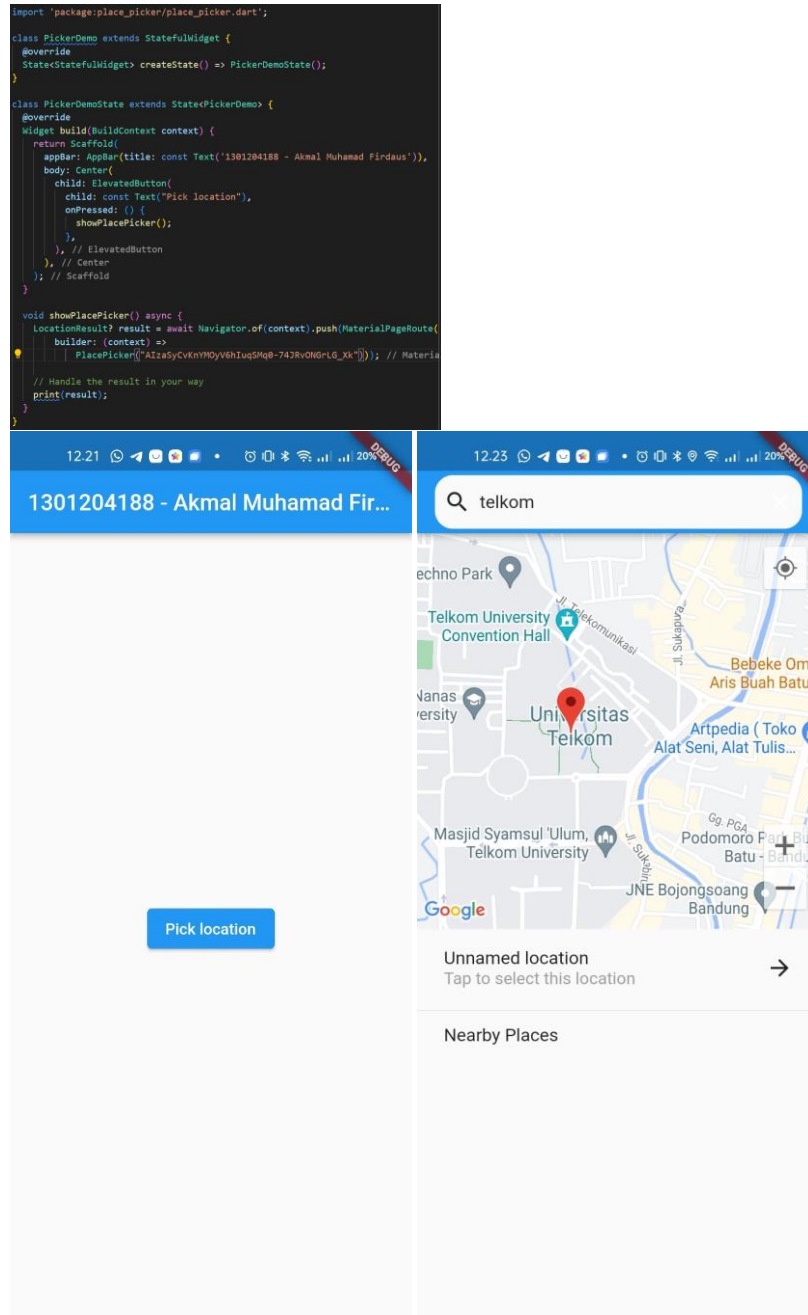
Place Picker adalah komponen atau fitur yang disediakan oleh Google Maps Platform untuk Flutter yang memungkinkan pengguna memilih lokasi dari peta dengan antarmuka yang intuitif. Dengan menggunakan Place Picker, pengguna dapat memilih lokasi dengan mengeklik atau mengetuk pada peta, dan mendapatkan informasi seperti alamat, koordinat geografis, dan detail lainnya tentang lokasi tersebut.

Untuk menggunakan Place Picker ikuti langkah-langkah berikut:

1. Tambahkan depedensi place_picker pada pubspec.yaml

```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.2  
  google_maps_flutter: ^2.2.8  
  permission_handler: ^10.2.0  
  place_picker: ^0.10.0
```

2. Gunakan codingan berikut untuk menampilkan place picker



3. Camera API

Camera API adalah sebuah plugin yang menyediakan fungsionalitas untuk mengakses kamera perangkat secara langsung. Dengan menggunakan Camera API, kita dapat menampilkan tampilan pratinjau dari kamera, mengambil gambar, merekam video, dan mengakses fitur-fitur lain yang tersedia pada kamera perangkat.

Untuk dapat menggunakan Camera pada aplikasi flutter ikut langkah-langkah berikut:

1. Tambahkan depedensi camera pada pubspec.yaml

```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.2  
  google_maps_flutter: ^2.2.8  
  permission_handler: ^10.2.0  
  place_picker: ^0.10.0  
  camera: ^0.10.5+2
```

2. Buat codingan sebagai berikut

```
import 'dart:async';
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';

List<CameraDescription> cameras = [];

Run | Debug | Profile
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  cameras = await availableCameras();
  runApp(CameraApp());
}

class CameraApp extends StatefulWidget {
  @override
  _CameraAppState createState() => _CameraAppState();
}

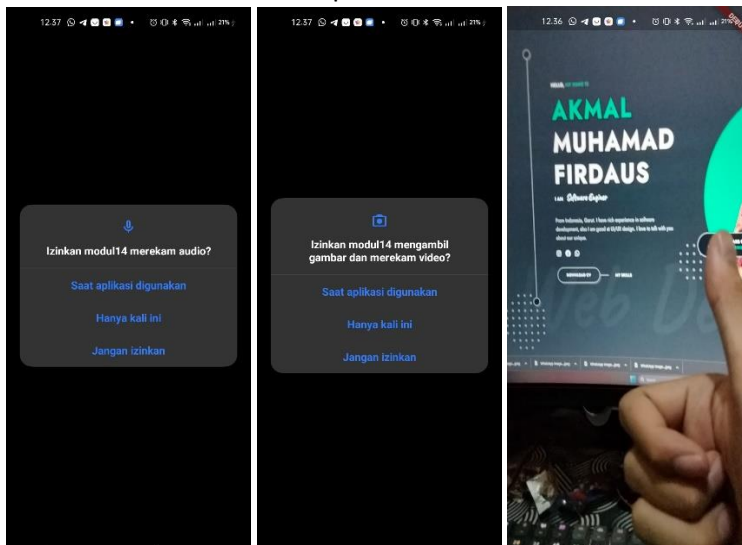
class _CameraAppState extends State<CameraApp> {
  late CameraController controller;

  @override
  void initState() {
    super.initState();
    controller = CameraController(cameras[0], ResolutionPreset.max);
    controller.initialize().then((_) {
      if (!mounted) {
        return;
      }
      setState(() {});
    });
  }

  @override
  void dispose() {
    controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (!controller.value.isInitialized) {
      return Container();
    }
    return MaterialApp(
      home: CameraPreview(controller),
    ); // MaterialApp
  }
}
```

Dikarenakan kamera membutuhkan sebuah permission dari sistem, codingan tersebut sudah secara otomatis akan meminta permission camera dan audio.



4. Media API

Media API pada adalah sekumpulan fungsi dan kelas yang digunakan untuk mengakses dan menggunakan fitur kamera atau galeri pada perangkat pengguna. Dengan menggunakan Media API, kita dapat mengambil gambar dari kamera, memilih gambar dari galeri, dan menampilkan gambar sebagai pratinjau atau mengirimnya ke server.

Untuk dapat menggunakan Media API ikuti langkah-langkah berikut:

1. Tambahkan depedensi image_picker kedalam pubspec.yaml

```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.2  
  google_maps_flutter: ^2.2.8  
  permission_handler: ^10.2.0  
  place_picker: ^0.10.0  
  camera: ^0.10.5+2  
  image_picker: ^0.8.7+5
```

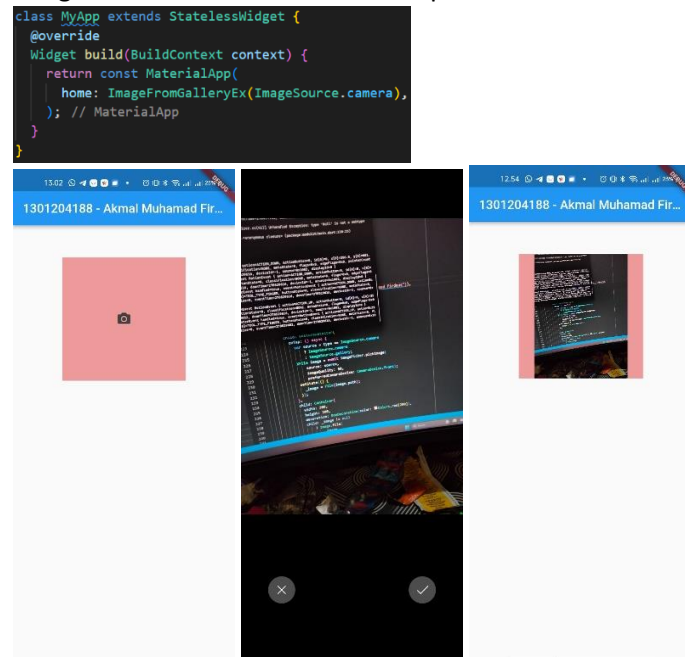
2. Tambahkan codingan sebagai berikut untuk menggunakan Media API

```
import 'dart:io';  
import 'package:flutter/material.dart';  
import 'package:image_picker/image_picker.dart';  
  
void main() => runApp(MyApp());  
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return const MaterialApp(  
      home: ImageFromGalleryEx(ImageSource.gallery),  
    );  
  }  
}  
  
class ImageFromGalleryEx extends StatefulWidget {  
  final type;  
  const ImageFromGalleryEx(this.type, {super.key});  
  @override  
  ImageFromGalleryExState createState() => ImageFromGalleryExState(type);  
}  
  
class ImageFromGalleryExState extends State<ImageFromGalleryEx> {  
  var _image;  
  var imagePicker;  
  var type;  
  ImageFromGalleryExState(this.type);  
  @override  
  void initState() {  
    super.initState();  
    imagePicker = ImagePicker();  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(title: Text("1301204188 - Akmal Muhammad Firdaus")),  
      body: Column(  
        children: <Widget>[  
          const SizedBox(  
            height: 52,  
          ),  
          Center(  
            child: GestureDetector(  
              onTap: () async {  
                var source = type == ImageSource.camera  
                  ? ImageSource.camera  
                  : ImageSource.gallery;  
                XFile image = await imagePicker.pickImage(  
                  source: source,  
                  imageQuality: 50,  
                  preferredCameraDevice: CameraDevice.front);  
                setState(() {  
                  _image = File(image.path);  
                });  
              },  
            child: Container(  
              width: 200,  
              height: 200,  
              decoration: BoxDecoration(color: Colors.red[200]),  
              child: _image != null  
                ? Image.file(  
                  _image,  
                  width: 200.0,  
                  height: 200.0,  
                  fit: BoxFit.fitHeight,  
                )  
                : Container(  
                  decoration: BoxDecoration(color: Colors.red[200]),  
                  width: 200,  
                  height: 200,  
                  child: Icon(  
                    Icons.camera_alt,  
                    color: Colors.grey[800],  
                  ),  
                ),  
            ),  
          ),  
        ],  
      ),  
    );  
  }  
}
```

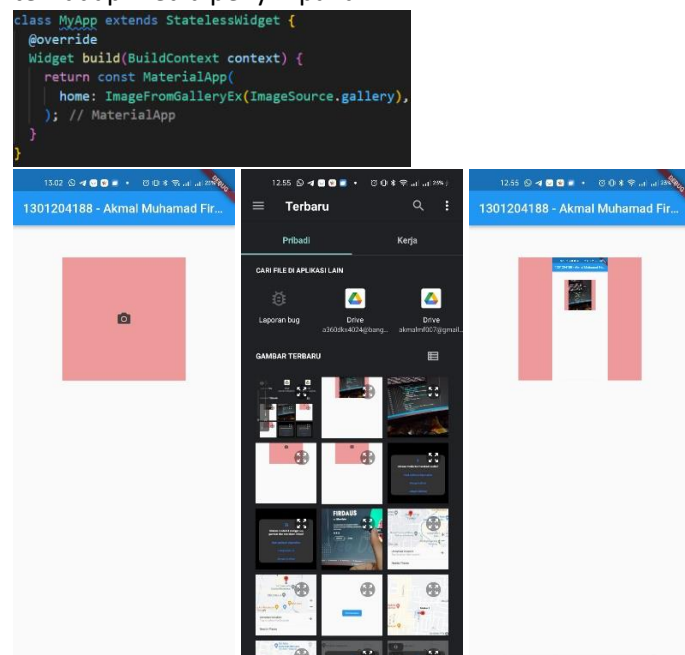

Terdapat 2 jenis Media API yang dapat digunakan, yaitu dari galery dan camera.

Pertama kita coba menggunakan Camera

1. Pada main function, ketika memanggil class ImageFromGalleryEx, tambahkan ImageSource.camera untuk mendapatkan akses terhadap camera



2. Untuk dapat menggunakan Media API Menggunakan galery, pada main function ketika memanggil class ImageFromGalleryEx, tambahkan ImageSource.gallery untuk mendapatkan akses terhadap media penyimpanan.



Jika sama sama dapat menggunakan camera, lantas apa perbedaan Media API dan camera pada poin 3 (Camera API). Pada Camera API, camera tertanam pada aplikasi flutternya langsung (embedded) jadi kita dapat menggunakan fitur camera lebih leluasa dan dapat lebih di customisasi seperti misalnya mengatur ukuran output, mengatur zoom, dll. Berbanding terbalik dengan Media API, API ini memanfaatkan komunikasi secara tidak langsung dari fitur camera yang ada pada sistem.

Full Source Code : <https://github.com/codernewbie04/CII3H4-ABP-Praktikum>