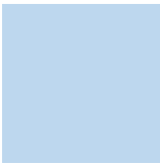


MODUL

PEMROGRAMAN BERBASIS PLATFORM



PERTEMUAN – 9

Camera & QR Scanner

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA





TUJUAN

Setelah menyelesaikan modul ini, praktikan diharapkan mampu :

1. Memahami penggunaan sintaks import library dalam modul Flutter.
2. Menggunakan fitur hardware kamera dalam pengembangan aplikasi.
3. Mengimplementasikan pemindaian (scanning) barcode, QR code.

DASAR TEORI

A. CAMERA

Popularitas kamera digital yang terus meningkat telah menginspirasi pengembang untuk menciptakan modul dan plugin Flutter yang memungkinkan akses ke fitur kamera pada perangkat Android dan iOS. Flutter menyediakan berbagai paket/plugin/library yang memungkinkan penggunaan dan kontrol kamera dalam aplikasi Flutter.

Dengan menggunakan modul Flutter yang sesuai, pengguna dapat mengakses dan mengontrol kamera perangkat langsung dari aplikasi yang dibangun dengan Flutter. Modul-modul ini memungkinkan pengambilan foto atau perekaman video dengan kamera perangkat, serta menampilkan pratinjau gambar yang ditangkap langsung ke pengguna.

Untuk mengintegrasikan fungsi kamera dalam aplikasi Flutter, Anda dapat menggunakan library Flutter yang spesifik untuk kamera. Plugin-plugin tersebut biasanya menyediakan API untuk mengakses fitur-fitur kamera, seperti mengambil foto, merekam video, mengatur pengaturan kamera, dan menampilkan pratinjau live dari kamera di dalam aplikasi.



B. QR SCANNER

Kita juga dapat memanfaatkan hardware kamera untuk mengembangkan aplikasi yang mampu melakukan scan barcode.

Dalam pengembangan aplikasi pemindai barcode, perlu mengintegrasikan library atau plugin yang menyediakan kemampuan pemindaian barcode dengan menggunakan kamera perangkat. Beberapa library populer seperti yang telah disebutkan sebelumnya, seperti `barcode_scan`, `qr_code_scanner`, atau `flutter_barcode_scanner`, dapat digunakan untuk mempermudah implementasi fitur pemindaian barcode dalam aplikasi Flutter Anda.

Setelah mendapatkan nilai kode dari pemindaian barcode, aplikasi dapat menggunakan informasi tersebut untuk berbagai tujuan, seperti menampilkan detail produk, memproses transaksi, membuka tautan web, atau menjalankan tindakan tertentu berdasarkan kode yang terdeteksi.

C. HARDWARE

Dengan interfaces/library/plugins tertentu kita dapat mengakses / memanfaatkan hardware dari perangkat kita. Contoh-contoh hardware dan sensor lainnya yang dapat Anda coba :

1. Camera - <https://pub.dev/packages/camera>
2. Geolocator - <https://pub.dev/packages/geolocator>
3. Sensor - https://pub.dev/packages/sensors_plus
4. Bluetooth - https://pub.dev/packages/flutter_blue_plus
5. Proximity - https://pub.dev/packages/proximity_sensor/example
6. Gyroscope
7. Accelerometer
8. Speaker



9. Dan lain-lain

***Teman-teman diharapkan mencari minimal 4++ hardware/sensor yang dapat diimplementasikan sesuai dengan topik tugas besar untuk mempermudah pengerjaan UGD !**

D. CAMERA API

Untuk membuat aplikasi yang dapat mengakses kamera, kita dapat menggunakan Api/library yang banyak tersedia di komunitas pengembang flutter. Referensi : <https://fluttergems.dev/camera/> . Pada modul ini, kita akan menggunakan beberapa library tambahan, yaitu :

```
1. camera: ^0.10.5+2 # access to hardware
2. mobile_scanner: ^3.2.0 # scan qr code
3. path_provider: ^2.0.15 # provide path to save file
4. qr_flutter: ^4.1.0 # generate qr code
```

GUIDED CAMERA, QR DAN BARCODE SCANNER

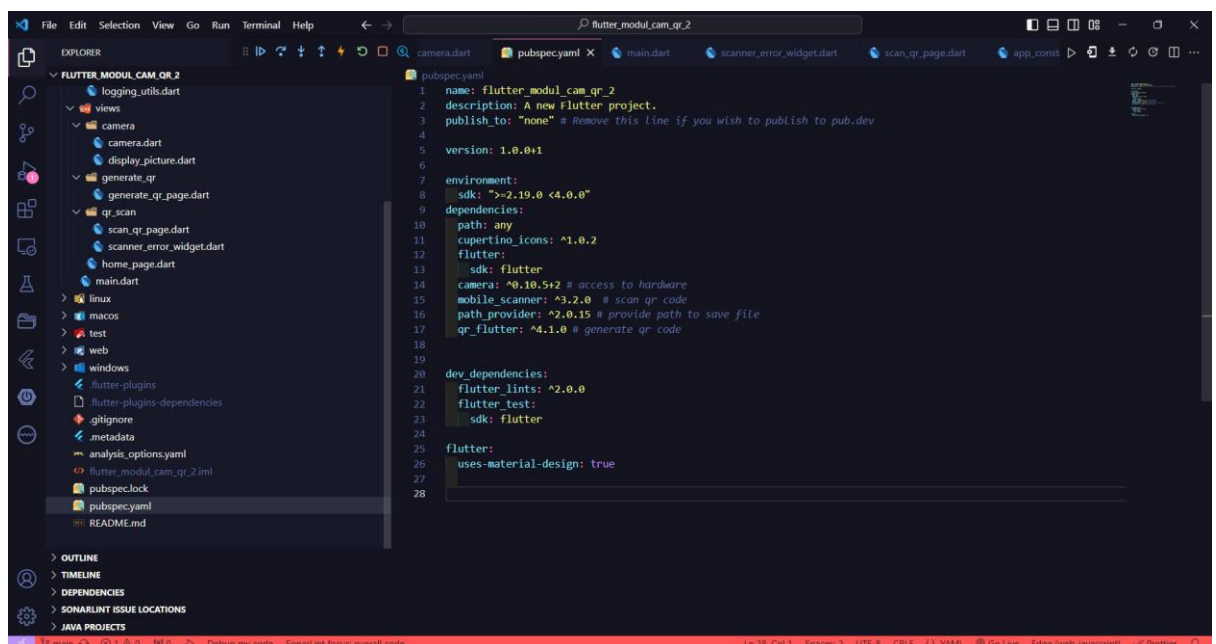
Poin yang dipelajari dalam Guided 1 ini, yaitu :

1. Link Video Demo : [20231025082040.mp4](https://www.youtube.com/watch?v=20231025082040)
2. Membuat sebuah aplikasi dengan 3 fungsionalitas utama:
 - a. Membuka dan mengambil gambar menggunakan camera
 - b. Generate QR code
 - c. Melakukan scan pada barcode

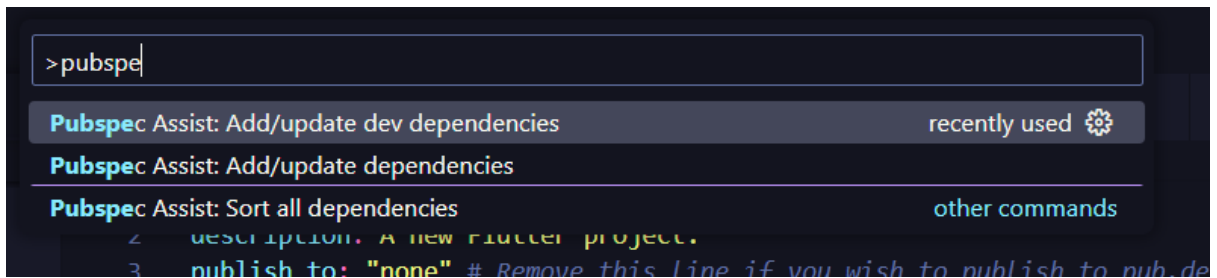
Silahkan ikuti langkah – langkah berikut ini :

1. Buat project dengan ketentuan sebagai berikut :
 - a. Application Name : modul_cam_qr_xxxx (x = 4 digit npm)
 - b. Disarankan saat membuat project memilih opsi empty project
2. Pada project kali ini kita akan memiliki 9 file :
 - a. app_constant.dart : representasi konstan yang akan kita gunakan di aplikasi. Ada 3 kelas yang berisi variable konstan statis yaitu
 - i. RouteConstant : Representasi route

- ii. ButtonTextConstant : Representasi String pada tombol-tombol yang digunakan
 - iii. LabelTextConstant : Representasi Text secara umum
 - b. logging_utils.dart : Merupakan sebuah kelas utility untuk templating log agar lebih rapi & informatif.
 - c. camera.dart : halaman UI yang menampilkan fungsionalitas mengambil gambar menggunakan hardware camera.
 - d. display_picture.dart : representasi tempat menampilkan gambar yang telah di capture sebelumnya
 - e. generate_qr_page.dart : halaman UI yang menampilkan fungsionalitas generate QR.
 - f. scan_qr_page.dart : halaman UI yang menampilkan fungsionalitas qr_scan
 - g. scanner_error_widget.dart : Sebuah kelas UI yang digunakan untuk handle error/kesalahan pada fungsionalitas qr_scan
 - h. Main.dart dan home_page.dart : halaman UI yang memiliki menu untuk mengarah ke fungsionalitas lainnya.
3. Sebelum kita masuk ke pengkodean, kita perlu mendaftarkan/mendefinisikan library yang akan kita gunakan di file pubspec.yaml (Harap Indentasi diperhatikan):



Gambar 1 pubspec.yaml dan struktur folder



Gambar 2 extension Pubspect Assist

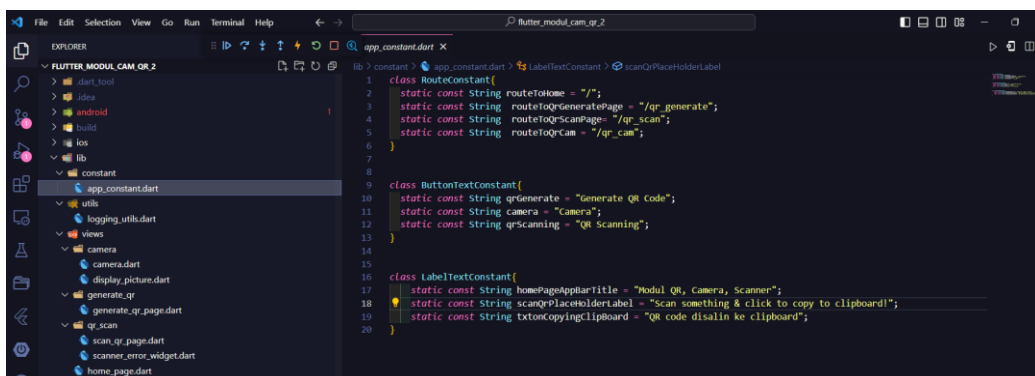
Anda dapat menambahkan dependency camera, mobile_scanner, path_provider dan qr_flutter. (Anda dapat menggunakan ekstensi pubspec assist untuk menghindari typo pada pengetikan dependency).

4. Selain itu, untuk menggunakan dependency camera Android sdk version di setting ke 21 pada android/app/build.gradle (perhatikan line 49)



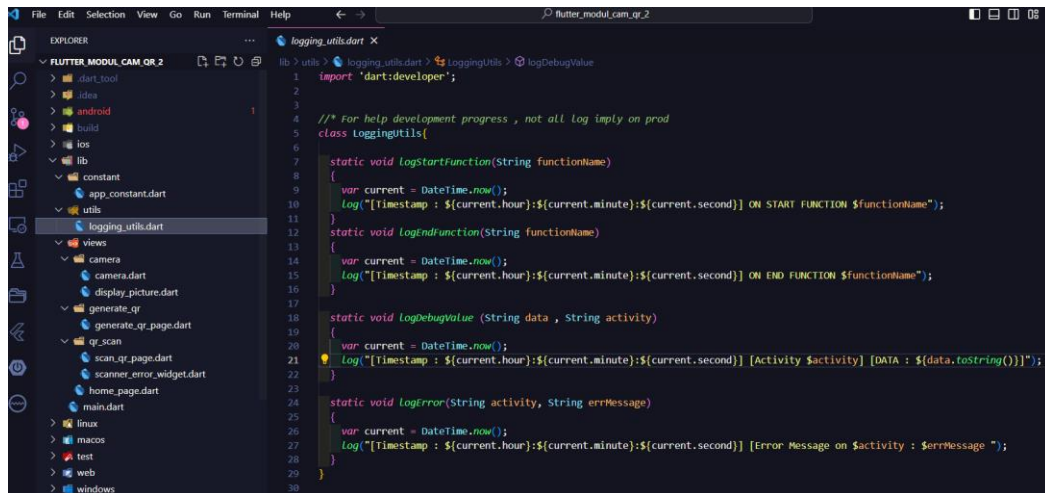
Gambar 3 Setting minSdkVersion

Pada file App_constant.dart, Anda dapat menyalin code berikut ini :



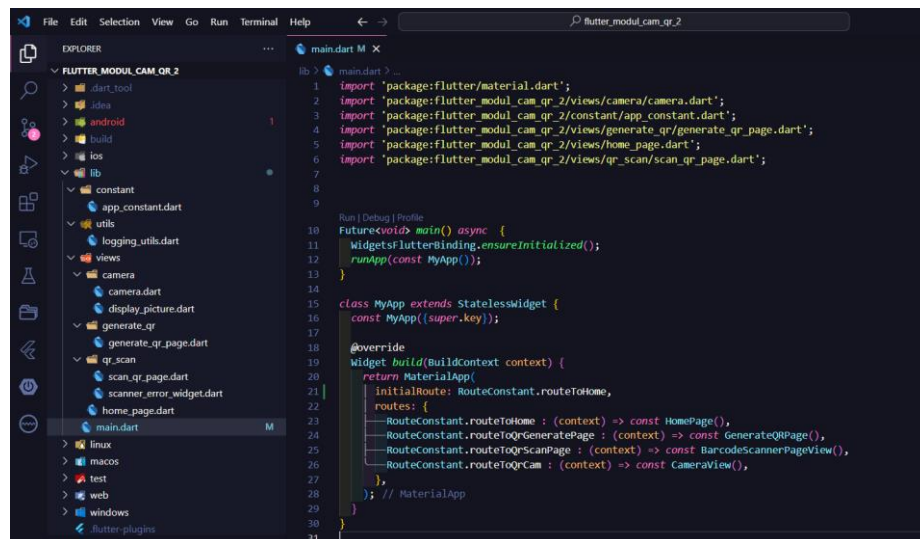
Gambar 4 AppConstant representasi konstanta yang digunakan pada aplikasi

5. Pada file logging_utils.dart, Anda dapat menyalin code berikut ini :



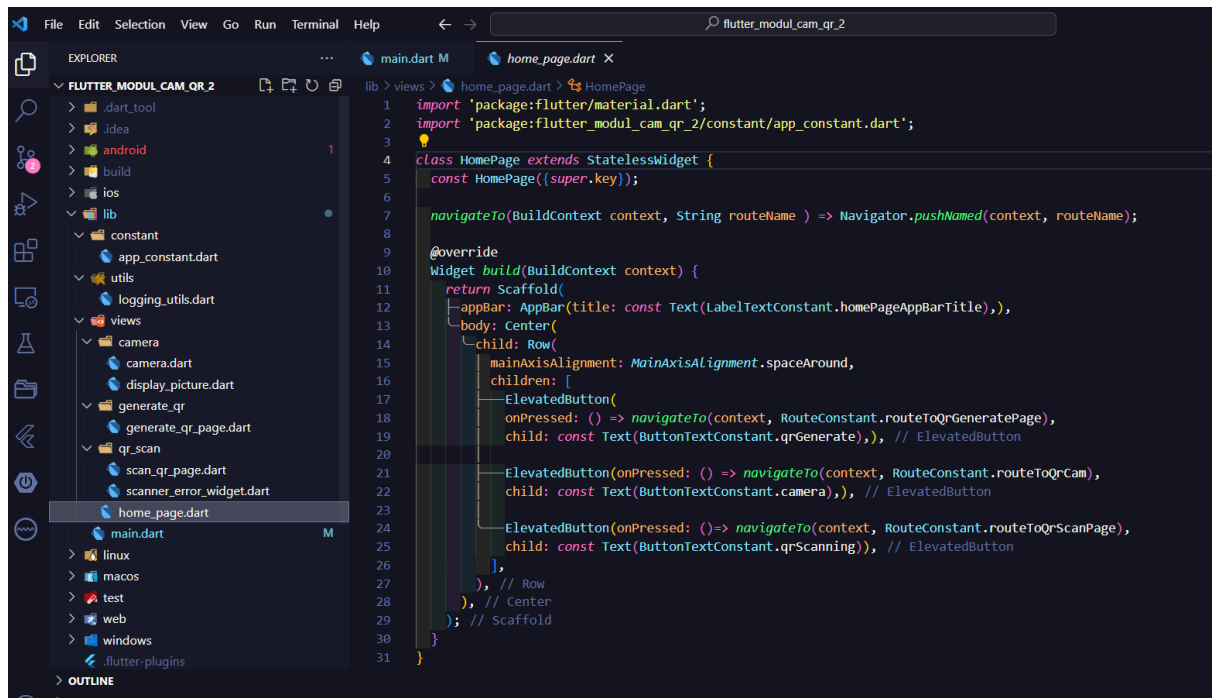
Gambar 5 Class LoggingUtils representasi utility untuk logging

6. Pada file main.dart , Anda dapat menyalin code berikut ini (pada step ini awalnya akan error karena view-view lainnya belum ada, bisa diabaikan dahulu) :



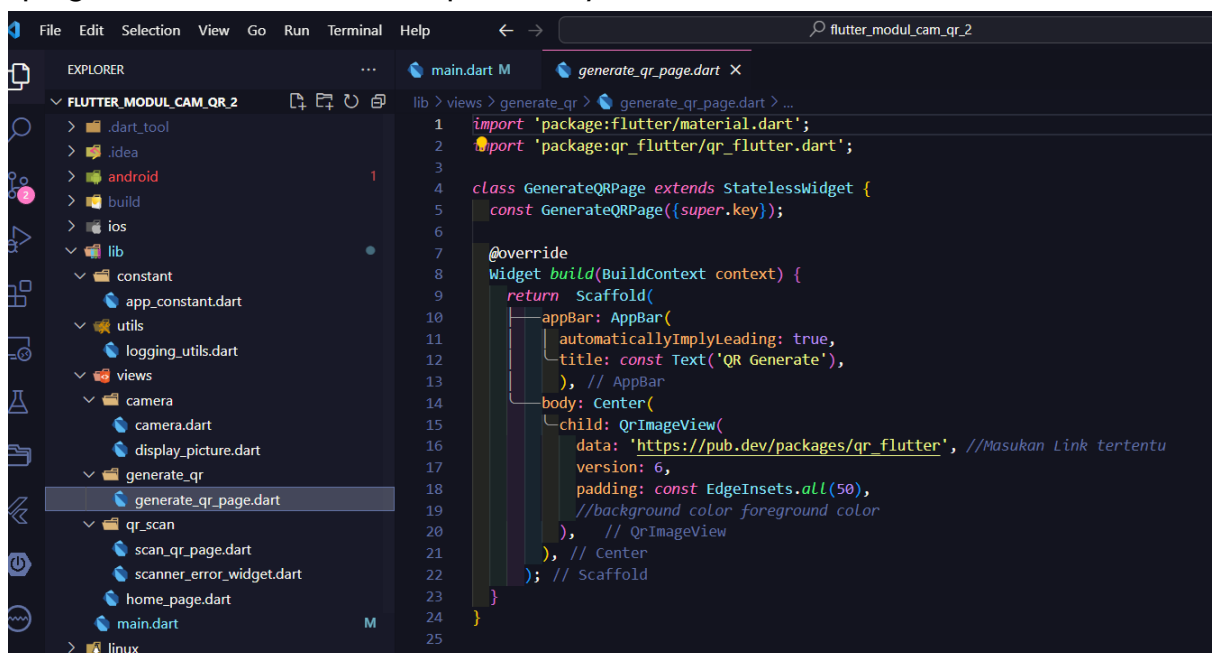
Gambar 6 File main.dart

7. Pada file home_page.dart, Anda dapat menyalin code berikut ini :



Gambar 7 File home_page.dart

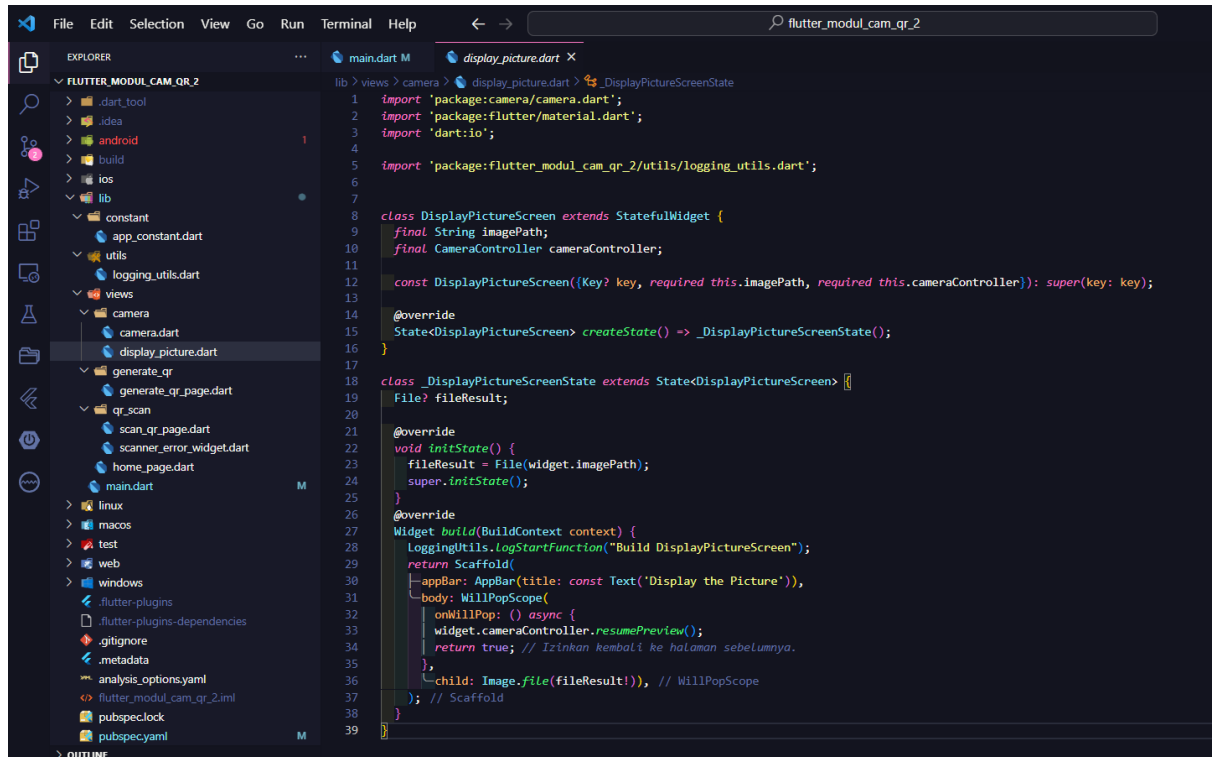
8. Kita mulai mengerjakan fungsionalitas generate QR. Pada file qr_generate.dart, Anda dapat menyalin code berikut ini :



Gambar 8 pengkodean pada file qr_generate.dart

9. Selanjutnya kita mengerjakan fungsionalitas mengambil gambar menggunakan camera dan menampilkannya . Pada fungsionalitas ini kita akan bekerja pada 2 file utama, yaitu camera.dart dan

display_picture.dart. Anda dapat menyalin code berikut ini untuk di file display_picture.dart:



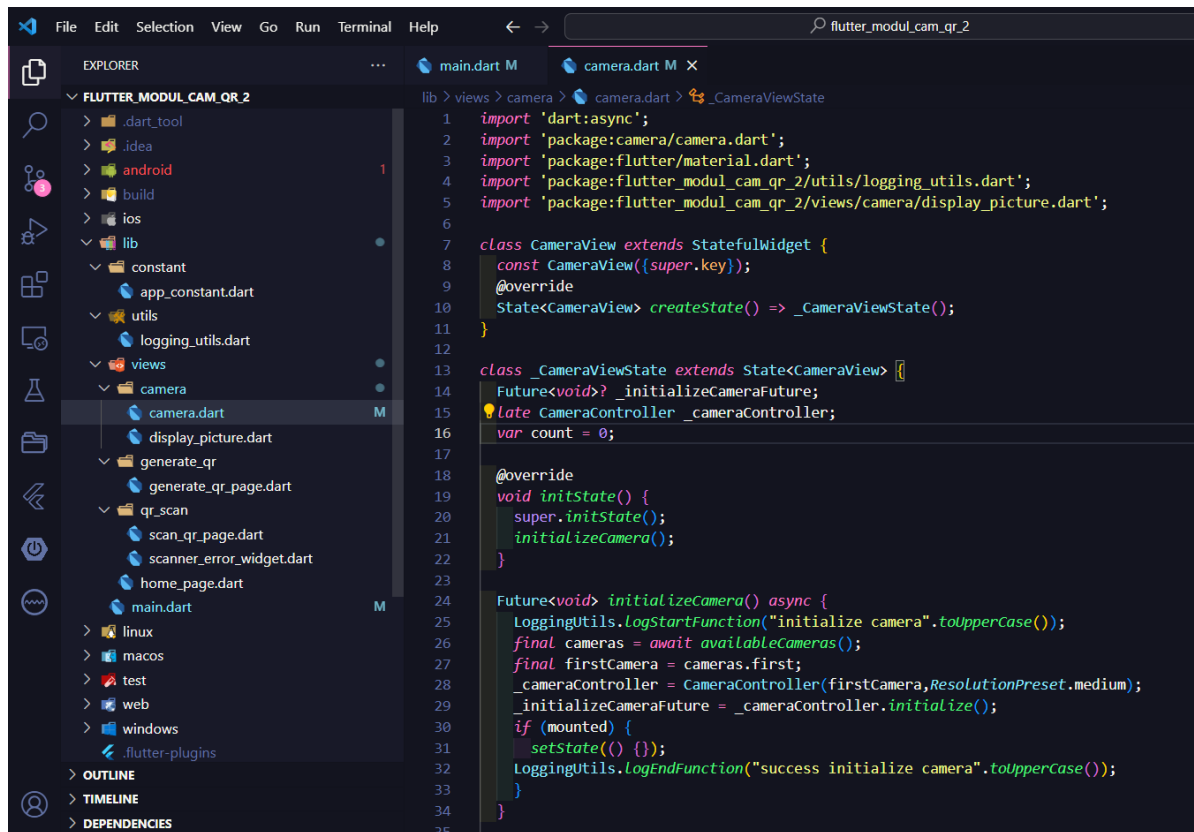
```

lib > views > camera > display_picture.dart > _DisplayPictureScreenState
1  import 'package:camera/camera.dart';
2  import 'package:flutter/material.dart';
3  import 'dart:io';
4
5  import 'package:flutter_modul_cam_qr_2/utills/logging_utils.dart';
6
7
8  class DisplayPictureScreen extends StatefulWidget {
9    final String imagePath;
10   final CameraController cameraController;
11
12   const DisplayPictureScreen({Key? key, required this.imagePath, required this.cameraController}): super(key: key);
13
14   @override
15   State<DisplayPictureScreen> createState() => _DisplayPictureScreenState();
16 }
17
18 class _DisplayPictureScreenState extends State<DisplayPictureScreen> {
19   File? fileResult;
20
21   @override
22   void initState() {
23     fileResult = File(widget.imagePath);
24     super.initState();
25   }
26
27   @override
28   Widget build(BuildContext context) {
29     LoggingUtils.logStartFunction("Build DisplayPictureScreen");
30     return Scaffold(
31       appBar: AppBar(title: const Text('Display the Picture')),
32       body: WillPopScope(
33         onWillPop: () async {
34           widget.cameraController.resumePreview();
35           return true; // Izinkan kembali ke halaman sebelumnya.
36         },
37         child: Image.file(fileResult!), // WillPopScope
38       ); // Scaffold
39   }

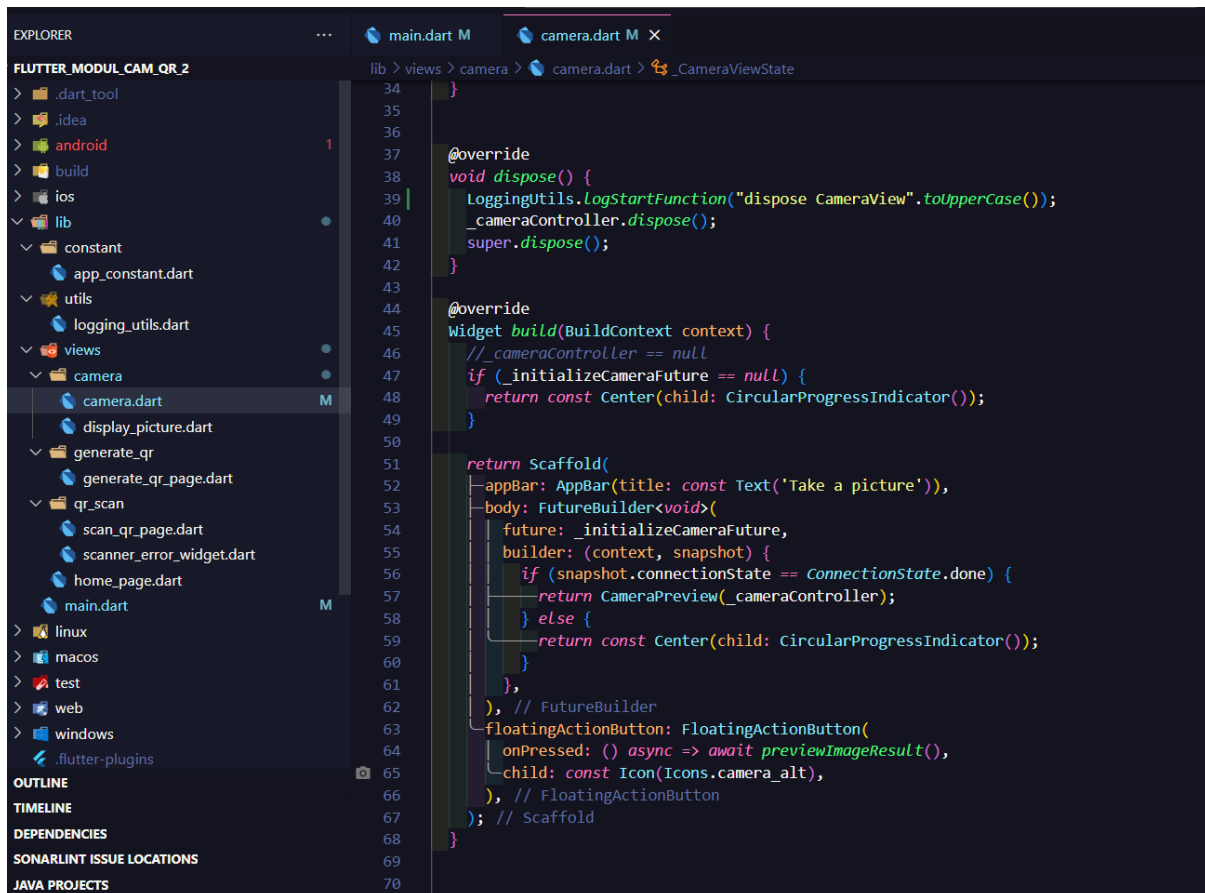
```

Gambar 9 File display_picture.dart

10. Pada file camera.dart, Anda dapat menyalin code berikut ini



Gambar 10 camera.dart bagian 1



Gambar 11 camera_dart bagian 2



Gambar 12 camera_dart bagian 3

11. Selanjutnya , kita mengerjakan fungsionalitas terakhir yaitu qr scan/ barcode scan yang pengkodeannya direpresentasikan pada scan_qr_page.dart dan scanner_error_widget.dart

```

lib > views > qr_scan > scan_qr_page.dart > _BarcodeScannerPageViewState > build
1  import 'package:flutter/material.dart';
2  import 'package:flutter/services.dart';
3  import 'package:flutter_module_cam_qr_2/constant/app_constant.dart';
4  import 'package:flutter_module_cam_qr_2/views/qr_scan/scanner_error_widget.dart';
5  import 'package:mobile_scanner/mobile_scanner.dart';
6
7  class BarcodeScannerPageView extends StatefulWidget {
8    const BarcodeScannerPageView({Key? key}) : super(key: key);
9
10   @override
11   State<BarcodeScannerPageView> createState() => _BarcodeScannerPageViewState();
12 }
13
14 class _BarcodeScannerPageViewState extends State<BarcodeScannerPageView>
15   with SingleTickerProviderStateMixin {
16   BarcodeCapture? barcodeCapture;
17
18   @override
19   Widget build(BuildContext context) {
20     return Scaffold(
21       backgroundColor: Colors.black,
22       body: PageView(
23         children: [
24           cameraView(),
25           container(),
26         ],
27       ), // PageView
28     ); // Scaffold
29   }
30 }

```

Gambar 13 Pengkodean scan_qr_page.dart bagian 1

```

lib > views > qr_scan > scan_qr_page.dart > _BarcodeScannerPageViewState > cameraView
31  Widget cameraView() {
32    return Builder(
33      builder: (context) {
34        return Stack(
35          children: [
36            MobileScanner(
37              startDelay: true,
38              controller: MobileScannerController(torchEnabled: false),
39              fit: BoxFit.contain,
40              errorBuilder: (context, error, child) {
41                return ScannerErrorWidget(error: error);
42              },
43              onDetect: (capture) => setBarcodeCapture(capture),
44            ), // MobileScanner
45            Align(
46              alignment: Alignment.bottomCenter,
47              child: Container(
48                alignment: Alignment.bottomCenter,
49                height: 100,
50                color: Colors.black.withOpacity(0.4),
51                child: Row(
52                  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
53                  children: [
54                    Center(
55                      child: SizedBox(
56                        width: MediaQuery.of(context).size.width - 120,
57                        height: 50,
58                        child: FittedBox(
59                          child: GestureDetector(
60                            onTap: () => getURLResult(),
61                            child: barcodeCaptureTextResult(context),
62                          ), // GestureDetector
63                        ), // FittedBox
64                      ), // SizedBox
65                    ), // Center
66                  ],
67                ), // Row
68              ), // Container
69            ), // Align
70          ],
71        ); // Stack
72      },
73    ); // Builder
74 }

```

Gambar 14 scan_qr_page.dart bagian 2 (CameraView)



```
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107
```

lib > views > qr_scan > scan_qr_page.dart > _BarcodeScannerPageViewState > cameraView

```
Text barcodeCaptureTextResult(BuildContext context) {  
  return Text(  
    barcodeCapture?.barcodes.first.rawValue ?? LabelTextConstant.scanQrPlaceholderLabel,  
    overflow: TextOverflow.fade,  
    style: Theme.of(context)  
      .textTheme  
      .headlineMedium!  
      .copyWith(color: Colors.white),  
  );  
}  
  
void setBarcodeCapture(BarcodeCapture capture) {  
  setState(() {  
    barcodeCapture = capture;  
  });  
}  
  
void getURLResult() {  
  final qrCode = barcodeCapture?.barcodes.first.rawValue;  
  if (qrCode != null) {  
    copyToClipboard(qrCode);  
  }  
}  
  
void copyToClipboard(String text) {  
  Clipboard.setData(ClipboardData(text: text));  
  ScaffoldMessenger.of(context).showSnackBar(  
    const SnackBar(content: Text(LabelTextConstant.txtonCopyingClipboard)),  
  );  
}
```

Gambar 15 scan_qr_page.dart bagian 3 (method & widget)



```
1 import 'package:flutter/material.dart';
2 import 'package:mobile_scanner/mobile_scanner.dart';
3
4
5 class ScannerErrorWidget extends StatelessWidget {
6   const ScannerErrorWidget({Key? key, required this.error}) : super(key: key);
7
8   final MobileScannerException error;
9
10  @override
11  Widget build(BuildContext context) {
12    String errorMessage;
13
14    switch (error.errorCode) {
15      case MobileScannerErrorCode.controllerUninitialized:
16        errorMessage = 'Controller not ready.';
17        break;
18      case MobileScannerErrorCode.permissionDenied:
19        errorMessage = 'Permission denied';
20        break;
21      default:
22        errorMessage = 'Generic Error';
23        break;
24    }
25
26    return ColoredBox(
27      | color: Colors.black,
28      child: Center(
29        | child: Column(
30          | mainAxisSize: MainAxisSize.min,
31          | children: [
32            | const Padding(
33              | padding: EdgeInsets.only(bottom: 16),
34              | child: Icon(Icons.error, color: Colors.white),
35            ), // Padding
36            | Text(
37              | errorMessage,
38              | style: const TextStyle(color: Colors.white),
39            ), // Text
40            | Text(
41              | error.errorDetails?.message ?? '',
42              | style: const TextStyle(color: Colors.white),
43            ), // Text
44          ],
45        ), // Column
46      ), // Center
47    ); // ColoredBox
48  }
49 }
```

Gambar 16 file scanner_error_widget.dart



PEMBAHASAN GUIDED QR, Cam, Scanner

1. Pada guided QR, Cam, Scanner ini. Kita akan membuat project dengan 3 fungsionalitas utama, yaitu :
 - a. Generate QR
 - b. Mengambil Gambar menggunakan Camera
 - c. Melakukan Scan QR
2. Ketiga fungsionalitas diatas, nantinya akan dapat diakses melalui halaman utama yang diimplementasikan dalam kelas 'HomePage'.
3. Untuk mencapai ketiga fungsionalitas tersebut, kita membutuhkan bantuan library atau dependency untuk mempermudah Pembangunan aplikasi mobile. Berikut penjelasan singkat library-library yang digunakan :
 - a. Camera : Memungkinkan aplikasi flutter untuk berinteraksi dengan kamera (jembatan/antarmuka/interface antara aplikasi dengan hardware camera)
 - b. Mobile_scanner : Library yang memungkinkan aplikasi flutter mengakses kamera dan menguraikan data QR /Barcode
 - c. Path_Provider : Membantu aplikasi flutter mengakses direktori / penyimpanan perangkat. (Akses data secara lokal)
 - d. qr_Flutter : memungkinkan Anda untuk membuat dan menampilkan QR dalam aplikasi Anda.
4. Untuk fungsionalitas generate QR Code, kita menggunakan library qr_flutter. Tepatnya kita menggunakan fungsi / widget QRImageview. Widget ini memungkinkan kita menampilkan gambar QR dari data tertentu. Detail dari widget ini, dapat teman-teman akses melalui link berikut ini :
https://pub.dev/documentation/qr_flutter/latest/
5. Untuk fungsionalitas mengambil gambar menggunakan kamera, kita menggunakan library camera, dart:async, dart:io. Dart async untuk mendukung jalannya fungsi secara asynchronous dan



dart:io berkaitan dengan proses input output. Selain itu, sering kali untuk membangun fungsionalitas mengambil gambar dikombinasikan dengan path provider untuk menyimpan gambar di lokasi tertentu (namun pada guided ini belum diterapkan).

6. Didalam `_cameraViewState`, terdapat beberapa variable penting:
 - a. `_InitializeCameraFuture` : variable yang digunakan untuk inisialisasi camera
 - b. `_cameraController` : Variabel untuk mengendalikan kamera yang telah diinisialisasi
 - c. Pada code, juga terdapat metode `initializeCamera()` yang nantinya akan dipanggil saat `initState()`. Metode ini menggunakan `availableCamera()` untuk mendapatkan daftar kamera yang tersedia pada perangkat dan menginisialisasikan `_cameraController` dengan kamera pertama yang terdapat pada perangkat
7. Dalam `build()`, Kita menggunakan `FutureBuild` untuk membangun tampilan berdasarkan status `_initializeCameraFuture`. Jika inisialisasi selesai (`ConnectionState.done`), `CameraPreview` akan menampilkan tampilan langsung dari kamera.
8. `FloatingActionButton` digunakan untuk mengambil foto. Saat tombol ditekan, Kita mengecek apakah inisialisasi kamera sudah selesai (`await _initializeCameraFuture`). Jika ya, kita menggunakan `_cameraController.takePicture()` untuk mengambil gambar.
9. Setelah gambar diambil, Kita membuka tampilan baru (`DisplayPictureScreen`) dan meneruskan path gambar yang diambil sebagai parameter.
10. `DisplayPictureScreen` adalah tampilan kedua yang menampilkan gambar yang diambil. Ini adalah `StatelessWidget` yang menerima path gambar sebagai parameter dan menampilkannya menggunakan `Image.file(File(imagePath))`.
11. Di bagian atas, kita mengimpor pustaka yang diperlukan:
 - a. `flutter/material.dart`,



- b. `mobile_scanner/mobile_scanner.dart` : Menyediakan kamera dan fungsi scanner
 - c. `flutter/services.dart` : digunakan untuk menyediakan service menyalin teks ke clipboard (copy teks)
12. Kita mendefinisikan kelas `BarcodeScannerPageView`, yang merupakan `StatefulWidget` yang menampilkan tampilan pemindaian kode QR atau barcode.
 13. Di dalam `_BarcodeScannerPageViewState`, kita mendeklarasikan variabel `capture` untuk menyimpan hasil pemindaian.
 14. Metode `cameraView()` digunakan untuk mengembalikan tampilan kamera yang menggunakan widget `MobileScanner` dari library `mobile_scanner`.
 15. Di dalam widget `MobileScanner`, kita mengatur parameter seperti `startDelay`, `controller`, dan lainnya untuk mengkonfigurasi perilaku pemindaian.
 16. Metode `onDetect` dijalankan ketika pemindaian berhasil dan hasilnya diterima. Di dalamnya, kita mengubah state untuk memperbarui `capture` dengan hasil pemindaian.
 17. Di dalam widget `Align`, kita menampilkan tampilan yang muncul di bagian bawah tampilan kamera. Di sini, kita menampilkan hasil pemindaian dan juga memberikan opsi untuk menyalin kode hasil pemindaian ke clipboard.
 18. Metode `copyToClipboard()` digunakan untuk menyalin teks (hasil pemindaian) ke clipboard dan menampilkan pesan ke pengguna menggunakan `SnackBar`.
 19. Metode `build()` digunakan untuk mengatur antarmuka aplikasi. Kita menggunakan `PageView` untuk menampilkan halaman pemindaian kamera dan halaman lainnya (kosong).
 20. Kelas `ScannerErrorWidget` adalah widget yang menangani pesan kesalahan yang mungkin terjadi selama pemindaian. Ini akan menampilkan pesan kesalahan yang sesuai berdasarkan kode kesalahan yang diterima dari `MobileScannerException`.



ATURAN Pengerjaan Guided :

- Guided dikerjakan selama waktu perkuliahan berlangsung.
- Penamaan projek guided harus sesuai dengan yang sudah dicontohkan.
- Guided dikumpulkan melalui github dengan penamaan setiap file pada github adalah : *guided_hardware_npm* (contoh : *guided_hardware_9999*)
 - **NAMAGUIDED → SESUAI DENGAN CONTOH DALAM MODUL INI**
 - **XXXX → 4 DIGIT TERAKHIR NPM**
- Setelah diupload melalui github, jangan lupa untuk mengumpulkan keseluruhan link file github melalui situs kuliah.
- Cara upload ke github, silahkan melihat pada modul **"UPLOAD GITHUB"**.