LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL 9

API PERANGKAT KERAS

Disusun Oleh :

Aditya Prabu Mukti / 2211104037

SE-06-02

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK FAKULTAS INFORMATIKA

2024

GUIDED

1. BAGIAN LOGIC

A. BUILD GRADLE Sourcecode android/app

```
plugins {
    id "com.android.application"
    id "kotlin-android"
    // The Flutter Gradle Plugin must be applied after the Android and Kotlin Gradle plugins.
    id "dev.flutter.flutter-gradle-plugin"
}

android {
    namespace = "com.example.pertemuan91"
    compileSdk = flutter.compileSdkVersion
    ndkVersion = flutter.ndkVersion

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = JavaVersion.VERSION_1_8
    }

    defaultConfig {
        // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
        applicationId = "com.example.pertemuan91"
        // You can update the following values to match your application needs.
        // For more information, see: https://flutter.dev/to/review-gradle-config.
        minSdkVersion 21
        targetSdk = flutter.targetSdkVersion
        versionCode = flutter.versionCode
        versionName = flutter.versionName
    }

    buildTypes {
        release {
            // TODO: Add your own signing config for the release build.
            // Signing with the debug keys for now, so `flutter run --release` works.
            signingConfig = signingConfigs.debug
        }
    }
}

flutter {
    source = "../.."
}
```

Deskripsi Program

File build.gradle digunakan untuk mengatur konfigurasi proyek, termasuk ID aplikasi, versi SDK yang digunakan seperti compileSdk, minSdkVersion, dan targetSdkVersion, serta pengaturan Gradle untuk Kotlin dan Android. Plugin Gradle Flutter diintegrasikan untuk mendukung fitur Flutter dalam proyek Android. Selain itu, konfigurasi mencakup pengaturan opsi JVM untuk Kotlin serta pengaturan tipe build rilis dengan debug signing untuk mendukung proses pengembangan. Direktori Flutter ditentukan melalui properti flutter { source }, yang menunjukkan lokasi utama source code proyek.

B. ANDROIDMANIFEST.XML

Sourcecode

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <application
        android:label="pertemuan91"
        android:name="${applicationName}"
        android:icon="@mipmap/ic_launcher">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTop"
            android:taskAffinity=""
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|locale|layoutDirec
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <!-- Specifies an Android theme to apply to this Activity as soon as
                 the Android process has started. This theme is visible to the user
                 while the Flutter UI initializes. After that, this theme continues
                 to determine the Window background behind the Flutter UI. -->
            <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>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate GeneratedPluginRegistrant.java -->
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
    <!-- Required to query activities that can process text, see:
         https://developer.android.com/training/package-visibility and
         https://developer.android.com/reference/android/content/Intent#ACTION_PROCESS_TEXT.

         In particular, this is used by the Flutter engine in io.flutter.plugin.text.ProcessTextPlugin. -->
    <queries>
        <intent>
            <action android:name="android.intent.action.PROCESS_TEXT"/>
            <data android:mimeType="text/plain"/>
        </intent>
    </queries>
</manifest>
```

Deskripsi program

File AndroidManifest.xml digunakan untuk mengatur izin, fitur, dan konfigurasi utama dari aplikasi Android berbasis Flutter. Aplikasi ini memerlukan izin akses kamera dengan deklarasi android.permission.CAMERA serta menyatakan kebutuhan perangkat keras kamera. Aktivitas utama, yaitu MainActivity, ditetapkan sebagai titik awal aplikasi dengan intent filter MAIN dan LAUNCHER. Pengaturan tema dan konfigurasi layar dirancang untuk memberikan pengalaman pengguna yang optimal, seperti dukungan rotasi layar dan akselerasi hardware. Metadata khusus disertakan untuk integrasi Flutter, termasuk pengaturan tema saat aplikasi dimulai dan deklarasi penggunaan Flutter embedding versi 2. Selain itu, elemen queries memungkinkan aplikasi memproses teks melalui intent eksternal.

C. PUBSPEC.YAML

Sourcecode

```yaml
name: pertemuan9_2
description: "A new Flutter project."
# The following line prevents the package from being accidentally published to
# pub.dev using `flutter pub publish`. This is preferred for private packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as versionCode.
# Read more about Android versioning at https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.
# Read more about iOS versioning at
# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build suffix.
version: 1.0.0+1

environment:
  sdk: ^3.5.3

# Dependencies specify other packages that your package needs in order to work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below to
# the latest version available on pub.dev. To see which dependencies have newer
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  camera: ^0.11.0+2
  image_picker: ^1.1.2

dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
  flutter_lints: ^4.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  # assets:
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants", see
  # https://flutter.dev/to/resolution-aware-images

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/to/asset-from-package

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/to/font-from-package
```

Deskripsi Program

File pubspec.yaml berfungsi sebagai konfigurasi utama untuk proyek Flutter bernama pertemuan91. Di dalamnya tercantum deskripsi proyek, versi aplikasi (1.0.0+1), serta lingkungan SDK Dart yang kompatibel (^3.5.3). File ini juga mencakup daftar dependensi utama, seperti flutter, cupertino_icons, camera, dan image_picker, yang mendukung fitur seperti ikon iOS, akses kamera, dan pemilihan gambar. Pada bagian dev_dependencies, terdapat flutter_test untuk pengujian aplikasi dan flutter_lints guna memastikan kualitas kode tetap terjaga. Properti uses-material-design: true memastikan ketersediaan ikon Material Design. Selain itu, konfigurasi tambahan seperti aset atau font kustom dapat diatur dalam bagian yang relevan. Struktur ini dibuat untuk mendukung kemudahan pengembangan dan pengelolaan proyek Flutter secara efisien.

2. BAGIAN TAMPILAN APLIKASI

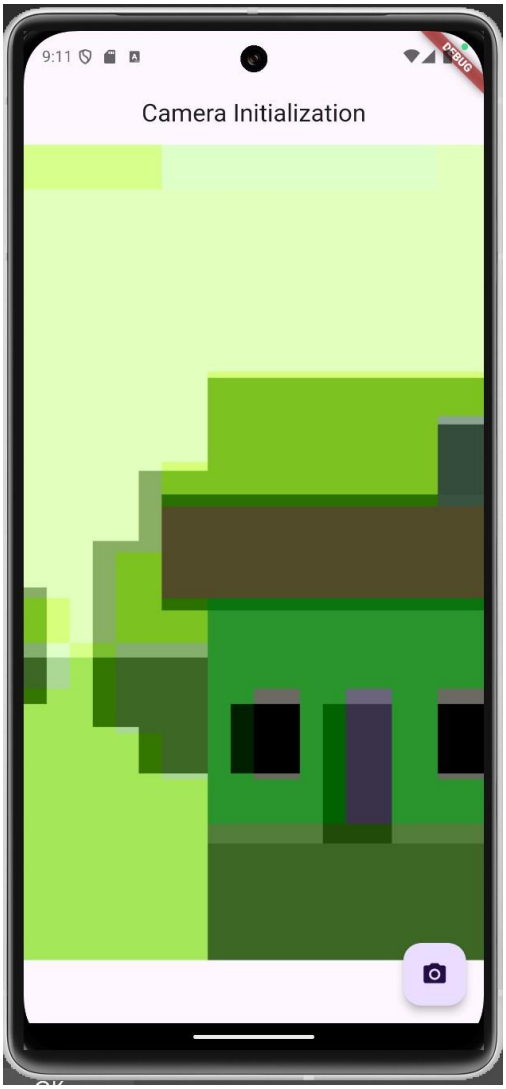
A. CAMERA SCREEN

Sourcecode

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

class CameraScreen extends StatefulWidget {
  const CameraScreen({super.key});

  @override
  _CameraScreenState createState() => _CameraScreenState();
}

class _CameraScreenState extends State<CameraScreen> {
  late CameraController _controller;
  Future<void>? _initializeControllerFuture; // Ubah late menjadi nullable

  @override
  void initState() {
    super.initState();
    _initializeCamera();
  }

  Future<void> _initializeCamera() async {
    // Ambil daftar kamera yang tersedia di perangkat
    final cameras = await availableCameras();
    final firstCamera = cameras.first;

    // Buat kontroler kamera dan mulai kamera
    _controller = CameraController(
      firstCamera,
      ResolutionPreset.high,
    );

    _initializeControllerFuture = _controller.initialize();
    setState(() {}); // Memperbarui UI setelah inisialisasi
  }

```

```dart
  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Camera Initialization'),
        centerTitle: true,
        backgroundColor: Colors.greenAccent[600],
      ), // AppBar
      body: FutureBuilder<void>(
        future: _initializeControllerFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.done) {
            return CameraPreview(_controller);
          } else {
            return const Center(child: CircularProgressIndicator());
          }
        },
      ), // FutureBuilder
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          try {
            await _initializeControllerFuture;

            final image = await _controller.takePicture();

            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (_) => DisplayPictureScreen(imagePath: image.path),
              ), // MaterialPageRoute
            );
          } catch (e) {
            print(e);
          }
        },
        child: const Icon(Icons.camera_alt),
      ), // FloatingActionButton
    ); // Scaffold
  }
}

class DisplayPictureScreen extends StatelessWidget {
  final String imagePath;

  const DisplayPictureScreen({super.key, required this.imagePath});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Display Picture'),
      ), // AppBar
      body: Center(
        child: Image.file(File(imagePath)),
      ), // Center
    ); // Scaffold
  }
}
```

Screenshoot output



Deskripsi Program

Kode ini mengimplementasikan fitur kamera dalam aplikasi Flutter menggunakan paket camera. Kelas CameraScreen berfungsi sebagai widget utama yang mengelola inisialisasi kamera perangkat menggunakan CameraController dengan pengaturan resolusi tinggi (ResolutionPreset.high). Inisialisasi kamera dilakukan dalam metode initState melalui fungsi _initializeCamera, yang mengambil daftar kamera yang tersedia pada perangkat dan memilih kamera pertama.

Antarmuka utama menggunakan widget FutureBuilder untuk memastikan kamera hanya ditampilkan setelah proses inisialisasi berhasil, dengan menampilkan pratinjau kamera melalui CameraPreview. Tombol aksi mengambang (FloatingActionButton) digunakan untuk menangkap gambar, menyimpannya, dan mengarahkan pengguna ke layar lain (DisplayPictureScreen) untuk menampilkan hasil foto melalui navigasi. Kelas DisplayPictureScreen bertugas menampilkan gambar yang telah diambil dengan memuat file gambar berdasarkan path yang diterima.

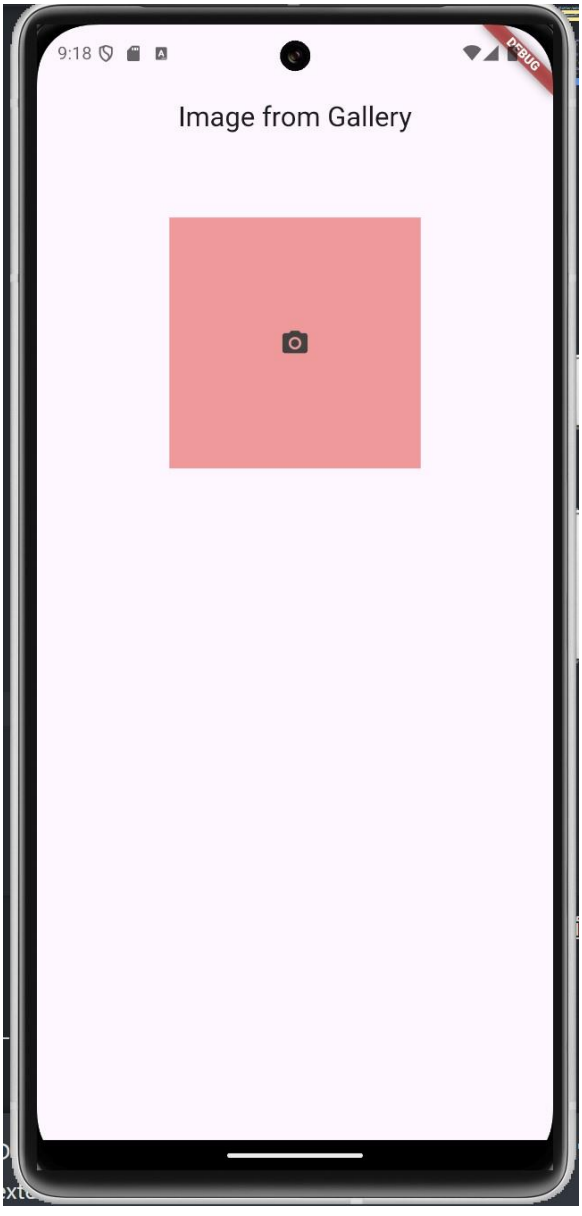
B. IMAGE PICKER SCREEN

Sourcecode

```dart
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';

enum ImageSourceType { gallery, camera }

class ImagePickerScreen extends StatefulWidget {
  final ImageSourceType type;

  const ImagePickerScreen({Key? key, required this.type}) : super(key: key);

  @override
  State<ImagePickerScreen> createState() => _ImagePickerScreenState();
}

class _ImagePickerScreenState extends State<ImagePickerScreen> {
  File? _image;
  late ImagePicker imagePicker;

  @override
  void initState() {
    super.initState();
    imagePicker = ImagePicker();
  }

  Future<void> _pickImage() async {
    // Pilih sumber gambar berdasarkan tipe yang diberikan
    final source = widget.type == ImageSourceType.camera
        ? ImageSource.camera
        : ImageSource.gallery;

    final pickedFile = await imagePicker.pickImage(
      source: source,
      imageQuality: 50, // Mengatur kualitas gambar
      preferredCameraDevice:
          CameraDevice.front, // Kamera depan jika menggunakan kamera
    );

    if (pickedFile != null) {
      setState(() {
        _image = File(pickedFile.path);
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(
          widget.type == ImageSourceType.camera
              ? "Image from Camera"
              : "Image from Gallery",
        ), // Text
        centerTitle: true,
      ), // AppBar
      body: Column(
        children: <Widget>[
          const SizedBox(height: 52),
          Center(
            child: GestureDetector(
              onTap: _pickImage,
              child: Container(
                width: 200,
                height: 200,
                decoration: BoxDecoration(
                  color: Colors.red[200],
                ), // BoxDecoration
                // Menampilkan gambar dari kamera atau galeri
                child: _image != null
                    ? Image.file(
                        image!,
```

```
                width: 200.0,
                height: 200.0,
                fit: BoxFit.fitHeight,
              ) // Image.file
            // Jika tidak ada gambar yang dipilih
            : Container(
                decoration: BoxDecoration(
                  color: Colors.red[200],
                ), // BoxDecoration
                width: 200,
                height: 200,
                child: Icon(
                  Icons.camera_alt,
                  color: Colors.grey[800],
                ), // Icon
              ), // Container
          ), // Container
        ), // GestureDetector
      ), // Center
    ], // <Widget>[]
  ), // Column
); // Scaffold
  }
}
```

Screenshoot Output



Deskripsi Program

Kode ini merupakan implementasi fitur pemilihan gambar dalam Flutter menggunakan paket image_picker. Kelas ImagePickerScreen menerima parameter type untuk menentukan sumber gambar, baik dari kamera maupun galeri, yang diatur menggunakan enum ImageSourceType. Saat aplikasi dijalankan, instance ImagePicker dibuat untuk memfasilitasi proses pemilihan gambar.
Metode _pickImage bertugas mengambil gambar sesuai sumber yang dipilih. Jika gambar berhasil diambil, path gambar disimpan dalam variabel _image, dan antarmuka diperbarui untuk menampilkan gambar tersebut. Jika belum ada gambar yang dipilih, aplikasi akan menampilkan ikon kamera sebagai tampilan default. Gambar yang dipilih ditampilkan dalam kontainer berukuran 200x200 piksel, dan pengguna dapat mengetuk kontainer tersebut untuk memulai proses pemilihan gambar ulang.

C. DISPLAY SCREEN

Sourcecode

```dart
import 'dart:io';
import 'package:flutter/material.dart';

class DisplayPictureScreen extends StatelessWidget {
  final String imagePath;

  const DisplayPictureScreen({Key? key, required this.imagePath})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Display Picture'),
        centerTitle: true,
        backgroundColor: Colors.greenAccent[600],
      ), // AppBar
      body: Center(
        child: Image.file(File(imagePath)),
      ), // Center
    ); // Scaffold
  }
}
```

Deskripsi Program

Kode ini adalah implementasi layar untuk menampilkan gambar yang dipilih atau diambil oleh pengguna menggunakan Flutter. Kelas DisplayPictureScreen merupakan widget stateless yang menerima path gambar melalui properti imagePath. Gambar ditampilkan menggunakan widget Image.file, yang memuat file berdasarkan path yang diberikan.
Antarmuka layar dilengkapi dengan AppBar yang memiliki judul "Display Picture" dan warna latar hijau muda. Gambar ditampilkan di tengah layar menggunakan widget Center untuk memastikan posisi gambar terpusat. Kode ini dirancang untuk menampilkan hasil gambar yang telah diambil atau dipilih oleh pengguna dalam aplikasi.

D. MAIN PROGRAM

Sourcecode

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan9_2/camera_screen.dart';
import 'package:pertemuan9_2/image_picker_screen.dart';
import 'package:image_picker/image_picker.dart';

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      home: const ImagePickerScreen(type: ImageSourceType.gallery),
    ); // MaterialApp
  }
}
```

Deskripsi Program

Program main ini merupakan titik awal aplikasi Flutter yang mengintegrasikan fitur kamera dan pemilihan gambar. Kelas MyApp bertindak sebagai widget utama yang menggunakan MaterialApp untuk mengatur tema aplikasi. Tema ini menggunakan skema warna ungu (Colors.deepPurple) dan mendukung Material Design 3 dengan properti useMaterial3: true. Properti home menentukan layar awal aplikasi, yang diatur ke CameraScreen, memungkinkan pengguna langsung mengakses fitur kamera. Program ini juga mengimpor file camera_screen.dart dan image_picker_screen.dart, memberikan fleksibilitas untuk memperluas aplikasi dengan layar lain, seperti ImagePickerScreen atau DisplayPictureScreen, sesuai dengan kebutuhan.

UNGUIDED

1. (Soal) Modifikasi project pemilihan gambar yang telah dikerjakan pada Tugas Pendahuluan Modul 09 agar fungsionalitas tombol dapat berfungsi untuk mengunggah gambar.
   - Ketika tombol Gallery ditekan, aplikasi akan mengambil gambar dari galeri, dan setelah gambar dipilih, gambar tersebut akan ditampilkan di dalam container.
   - Ketika tombol Camera ditekan, aplikasi akan mengambil gambar menggunakan kamera, dan setelah pengambilan gambar selesai, gambar tersebut akan ditampilkan di dalam container.
   - Ketika tombol Hapus Gambar ditekan, gambar yang ada pada container akan dihapus.

Sourcecode

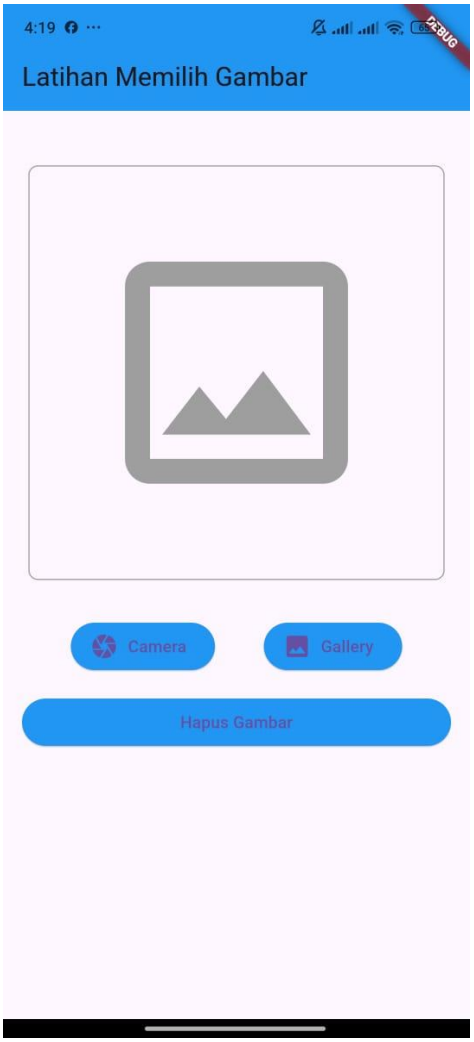
```dart
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';

Run | Debug | Profile
void main() {
  runApp(app: MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Latihan Memilih Gambar',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: ImagePickerPage(),
    ); // MaterialApp
  }
}

class ImagePickerPage extends StatefulWidget {
  @override
  _ImagePickerPageState createState() => _ImagePickerPageState();
}

class _ImagePickerPageState extends State<ImagePickerPage> {
  File? _imageFile;
  final ImagePicker _picker = ImagePicker();

  Future<void> _pickImageFromGallery() async {
    final XFile? pickedFile = await _picker.pickImage(source: ImageSource.gallery);
    if (pickedFile != null) {
      setState(fn: () {
        _imageFile = File(path: pickedFile.path);
      });
```

```
  }
}

Future<void> _pickImageFromCamera() async {
  final XFile? pickedFile = await _picker.pickImage(source: ImageSource.camera);
  if (pickedFile != null) {
    setState(fn: () {
      _imageFile = File(path: pickedFile.path);
    });
  }
}

void _clearImage() {
  setState(fn: () {
    _imageFile = null;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(data: 'Latihan Memilih Gambar'),
      backgroundColor: Colors.blue,
    ), // AppBar
    body: Padding(
      padding: const EdgeInsets.all(value: 16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: <Widget>[
          SizedBox(height: 30),
          Container(
            height: 350,
            width: 350,
            decoration: BoxDecoration(
              border: Border.all(color: Colors.grey),
              borderRadius: BorderRadius.circular(radius: 8),
            ), // BoxDecoration
            child: _imageFile != null
                ? Image.file(file: _imageFile!, fit: BoxFit.cover)
                : Icon(
                    icon: Icons.image_outlined,
                    size: 250,
                    color: Colors.grey,
                  ), // Icon
          ), // Container
          SizedBox(height: 32),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: <Widget>[
              ElevatedButton.icon(
                onPressed: _pickImageFromCamera,
                icon: Icon(icon: Icons.camera),
                label: Text(data: 'Camera'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                ),
              ), // ElevatedButton.icon
              ElevatedButton.icon(
                onPressed: _pickImageFromGallery,
                icon: Icon(icon: Icons.photo),
                label: Text(data: 'Gallery'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                ),
              ), // ElevatedButton.icon
            ],
          ), // Row
          SizedBox(height: 16),
          ElevatedButton(
```

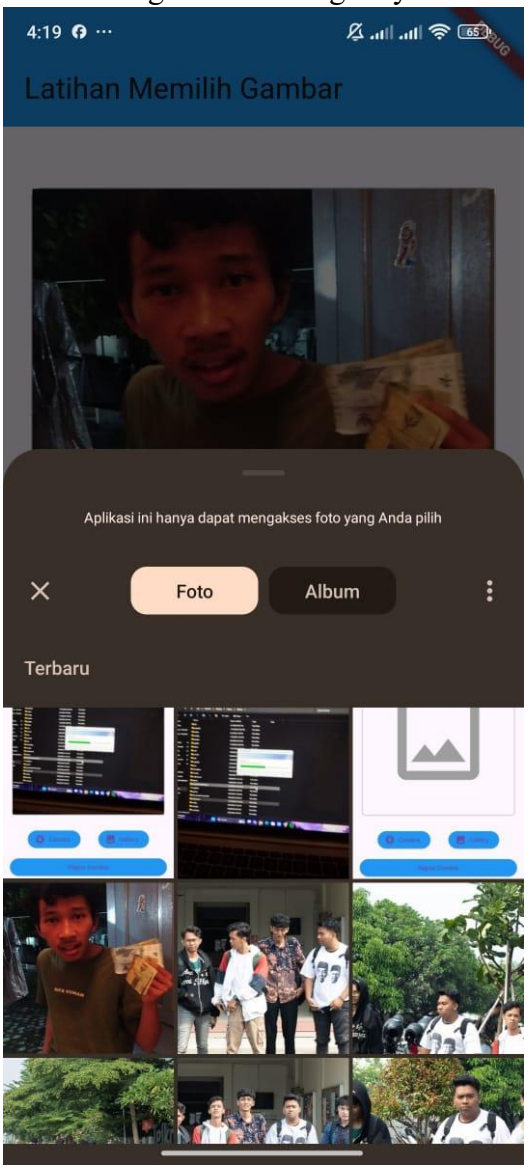
```
107                     onPressed: _clearImage,
108                     child: Text(data: 'Hapus Gambar'),
109                     style: ElevatedButton.styleFrom(
110                       backgroundColor: Colors.blue,
111                       minimumSize: Size(width: double.infinity, height: 40),
112                     ),
113                   ), // ElevatedButton
114                 ],
115               ), // Column
116             ), // Padding
117           ); // Scaffold
118         }
119       }
120
```

Screenshoot Output

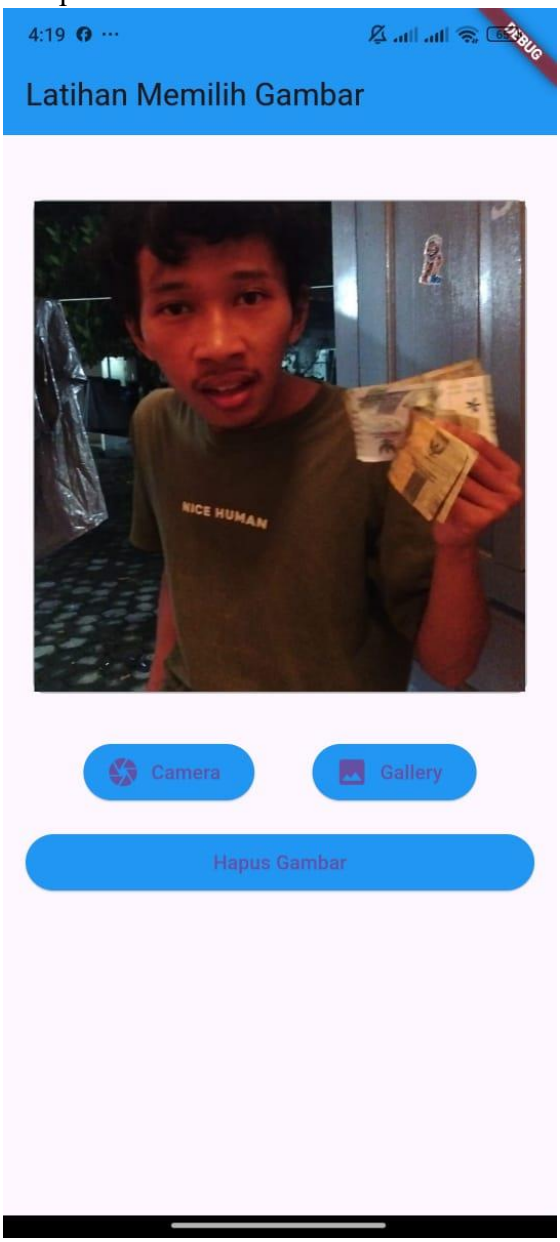
Tampilan awal:



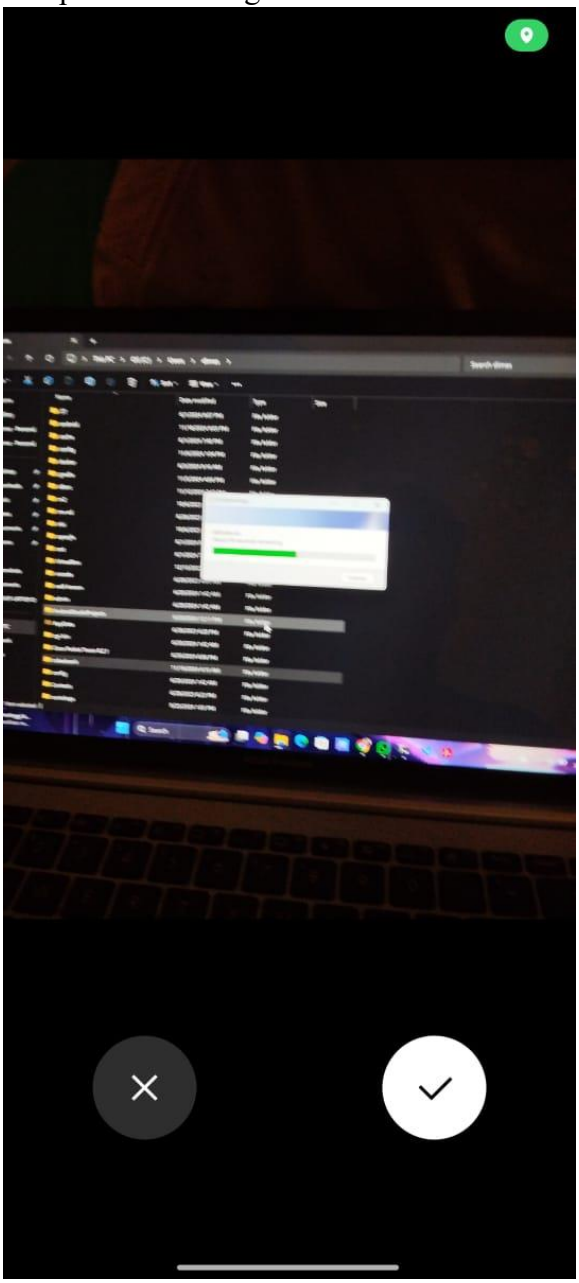
Tampilan saat meng klik tombol galery:



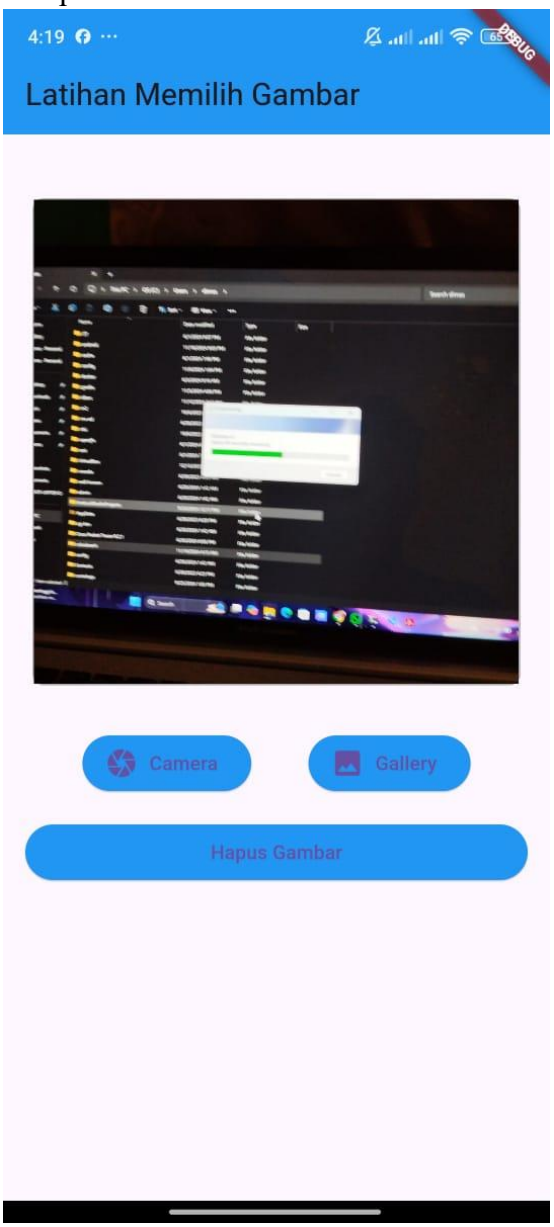
Tampilan saat sudah di ubah:



Tampilan saat meng klik tombol camera:



Tampilan saat sudah di ubah melalui kamera:



Deskripsi Program

Program ini adalah aplikasi Flutter yang memungkinkan pengguna memilih gambar melalui kamera atau galeri menggunakan package image_picker. Aplikasi ini terdiri dari widget utama MyApp, yang menampilkan halaman ImagePickerPage. Pada halaman ini, pengguna dapat melihat gambar yang dipilih dalam kotak pratinjau.
Pengguna dapat memilih gambar dari kamera atau galeri dengan menekan tombol yang sesuai. Selain itu, tersedia tombol tambahan untuk menghapus gambar yang telah dipilih. File gambar yang dipilih disimpan dalam variabel _imageFile dan akan ditampilkan menggunakan widget Image.file jika gambar tersedia.