LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL 9

API PERANGKAT KERAS

Disusun Oleh :

Dimas Cahyo Margono / 2211104060

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 ini mengatur konfigurasi proyek seperti ID aplikasi, versi SDK yang digunakan (compileSdk, minSdkVersion, targetSdkVersion), dan pengaturan Gradle untuk Kotlin dan Android. Plugin Flutter Gradle digunakan untuk mengintegrasikan Flutter dengan proyek Android, memungkinkan penggunaan fitur Flutter dalam aplikasi. Konfigurasi juga mencakup opsi JVM untuk Kotlin dan opsi build tipe rilis dengan debug signing untuk mempermudah pengembangan. Flutter directory diatur melalui properti flutter { source }, menunjukkan lokasi source code utama proyek.

B. ANDROIDMANIFEST.XML

Sourcecode

```
<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|smal
lestScreenSize|locale|layoutDirection|fontScale|screenLayout|density|uiMode"
            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` ini mengatur izin, fitur, dan konfigurasi utama aplikasi Android berbasis Flutter. Aplikasi ini membutuhkan akses kamera melalui izin `android.permission.CAMERA` dan mendeklarasikan kebutuhan perangkat keras kamera. Aktivitas utama adalah `MainActivity`, yang ditandai sebagai titik masuk aplikasi dengan intent filter `MAIN` dan `LAUNCHER`. Tema dan konfigurasi layar diatur untuk mendukung pengalaman pengguna yang optimal, seperti rotasi layar dan akselerasi hardware. Metadata khusus digunakan untuk integrasi Flutter, termasuk pengaturan tema saat proses awal dan deklarasi penggunaan Flutter embedding versi 2. Selain itu, elemen `queries` memungkinkan aplikasi memproses teks melalui intent eksternal.

C. PUBSPEC.YAML

Sourcecode

```
name: pertemuan91
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/InfoPlistK
eyReference/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
```

```yaml
# 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` ini mengatur konfigurasi proyek Flutter bernama `pertemuan91`, termasuk deskripsi, versi aplikasi (`1.0.0+1`), dan lingkungan SDK Dart (`^3.5.3`). File ini juga mendefinisikan dependensi utama seperti `flutter`, `cupertino_icons`, `camera`, dan `image_picker`, yang digunakan untuk menambahkan ikon iOS, mengakses kamera, dan memilih gambar. Bagian dev_dependencies menyertakan `flutter_test` untuk pengujian aplikasi dan `flutter_lints` untuk memastikan praktik pengkodean yang baik. Properti `uses-material-design: true` memastikan ikon Material tersedia, dan berbagai konfigurasi tambahan seperti penambahan aset atau font kustom dapat diatur di bagian yang sesuai. Konfigurasi ini dirancang untuk mempermudah pengembangan dan pengelolaan proyek Flutter.

2. BAGIAN TAMPILAN APLIKASI

Setelah kita mengkonfigurasi perizinan-perizinan kamera, kita tinggal merancang tampilan aplikasi beserta pengetesan kamera dan akses galeri untuk menandakan bahwa aplikasi beserta notifikasi berjalan dengan baik

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});
```

```dart
  @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
  }

  @override
  void dispose() {
    // Bersihkan kontroler ketika widget dihapus
    _controller.dispose();
    super.dispose();
  }

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

```
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          try {
            // Pastikan kamera sudah diinisialisasi
            await _initializeControllerFuture;

            // Ambil gambar
            final image = await _controller.takePicture();

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

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'),
      ),
      body: Center(
        child: Image.file(File(imagePath)),
      ),
    );
  }
}
```

Screenshoot output

Deskripsi Program

Kode ini merupakan implementasi fitur kamera dalam aplikasi Flutter menggunakan paket `camera`. Kelas `CameraScreen` adalah widget utama yang menginisialisasi kamera perangkat menggunakan `CameraController`, dengan resolusi tinggi (`ResolutionPreset.high`). Kamera diinisialisasi di dalam metode `initState` melalui `_initializeCamera`, yang mengambil daftar kamera perangkat dan memilih kamera pertama. Pada tampilan utama, widget `FutureBuilder` memastikan kamera hanya ditampilkan setelah berhasil diinisialisasi, dengan `CameraPreview` sebagai pratinjau kamera. Tombol mengambang (`FloatingActionButton`) digunakan untuk mengambil gambar, menyimpannya, dan menampilkan hasil pada layar lain (`DisplayPictureScreen`) melalui navigasi. Kelas `DisplayPictureScreen` menampilkan gambar yang diambil dengan memuat file gambar berdasarkan path yang diberikan

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",
        ),
        centerTitle: true,
      ),
      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],
                ),
                // Menampilkan gambar dari kamera atau galeri
                child: _image != null
                    ? Image.file(
                        _image!,
                        width: 200.0,
                        height: 200.0,
                        fit: BoxFit.fitHeight,
                      )
                    // Jika tidak ada gambar yang dipilih
                    : Container(
                        decoration: BoxDecoration(
                          color: Colors.red[200],
                        ),
                        width: 200,
                        height: 200,
                        child: Icon(
                          Icons.camera_alt,
                          color: Colors.grey[800],
                        ),
                      ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

Screenshoot Output

Deskripsi Program

Kode ini adalah implementasi fitur pemilihan gambar menggunakan paket `image_picker` di Flutter. Kelas `ImagePickerScreen` menerima parameter `type` untuk menentukan sumber gambar, yaitu dari kamera atau galeri, melalui enum `ImageSourceType`. Saat aplikasi dimulai, sebuah instance `ImagePicker` diinisialisasi. Metode `_pickImage` digunakan untuk memilih gambar berdasarkan sumber yang ditentukan. Jika gambar berhasil dipilih, path gambar disimpan dalam variabel `_image` dan UI diperbarui untuk menampilkan gambar tersebut. Jika belum ada gambar yang dipilih, tampilan default berupa ikon kamera akan ditampilkan. Gambar ditampilkan dalam kontainer berukuran 200x200 piksel, dan pengguna dapat mengetuknya untuk memulai proses pemilihan gambar lagi.

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],
      ),
      body: Center(
        child: Image.file(File(imagePath)),
      ),
    );
  }
}
```

Deskripsi Program

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

D. MAIN PROGRAM

Sourcecode

```dart
import 'package:flutter/material.dart';
import 'package:pertemuan91/camera_screen.dart';
import 'package:pertemuan91/image_picker_screen.dart';
import 'package:image_picker/image_picker.dart';

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,
      ),
      home: CameraScreen(),
      // DisplayPictureScreen()
    );
  }
}
```

Deskripsi Program

Program `main` ini adalah entry point aplikasi Flutter yang menampilkan fitur kamera dan pemilihan gambar. Kelas `MyApp` adalah widget utama yang menggunakan `MaterialApp` untuk mengatur tema aplikasi dengan skema warna berbasis warna ungu (`Colors.deepPurple`) dan mendukung Material Design 3 (`useMaterial3: true`). Properti `home` menentukan tampilan awal aplikasi, diatur ke `CameraScreen`, yang memungkinkan pengguna mengakses fitur kamera. Program ini juga mengimpor file `camera_screen.dart` dan `image_picker_screen.dart`, sehingga aplikasi dapat diperluas untuk menampilkan layar lain seperti `ImagePickerScreen` atau `DisplayPictureScreen` sesuai 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.

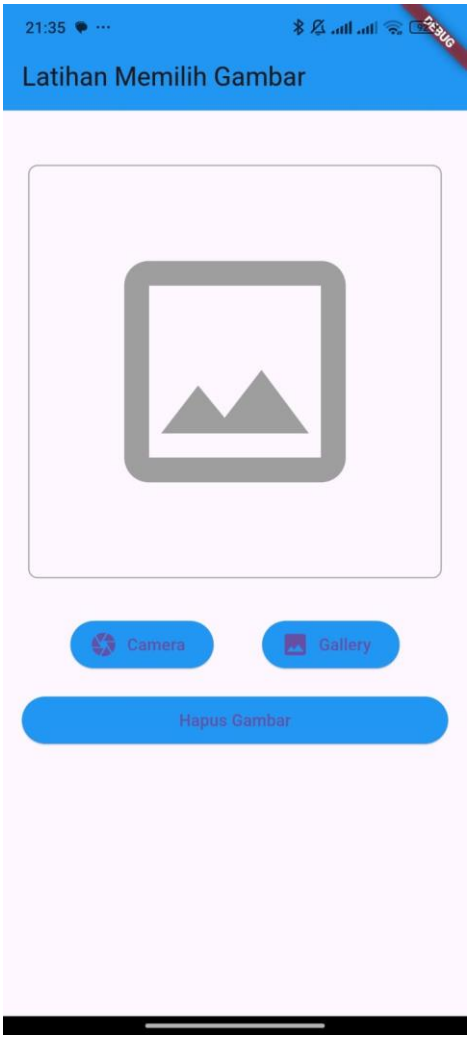
Sourcecode

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

void main() {
  runApp(MyApp());
}

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

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

class _ImagePickerPageState extends State<ImagePickerPage> {
  File? _imageFile;

  final ImagePicker _picker = ImagePicker();

  Future<void> _pickImageFromGallery() async {
    final pickedFile = await _picker.pickImage(source: ImageSource.gallery);
    if (pickedFile != null) {
```

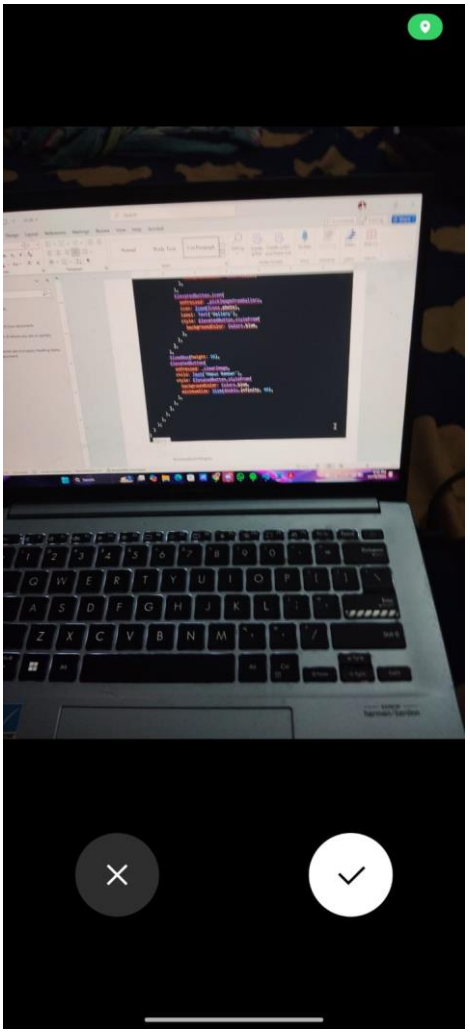
```
    setState(() {
      _imageFile = File(pickedFile.path);
    });
  }
}

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

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

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Latihan Memilih Gambar'),
      backgroundColor: Colors.blue,
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          SizedBox(height: 30),
          Container(
            height: 350,
            width: 350,
            decoration: BoxDecoration(
              border: Border.all(color: Colors.grey),
              borderRadius: BorderRadius.circular(8),
            ),
            child: _imageFile != null
                ? Image.file(_imageFile!, fit: BoxFit.cover)
                : Icon(
                    Icons.image_outlined,
                    size: 250,
                    color: Colors.grey,
                  ),
          ),
          SizedBox(height: 32),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
```

```
            children: [
              ElevatedButton.icon(
                onPressed: _pickImageFromCamera,
                icon: Icon(Icons.camera),
                label: Text('Camera'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                ),
              ),
              ElevatedButton.icon(
                onPressed: _pickImageFromGallery,
                icon: Icon(Icons.photo),
                label: Text('Gallery'),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.blue,
                ),
              ),
            ],
          ),
          SizedBox(height: 16),
          ElevatedButton(
            onPressed: _clearImage,
            child: Text('Hapus Gambar'),
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              minimumSize: Size(double.infinity, 40),
            ),
          ),
        ],
      ),
    ),
  );
 }
}
```

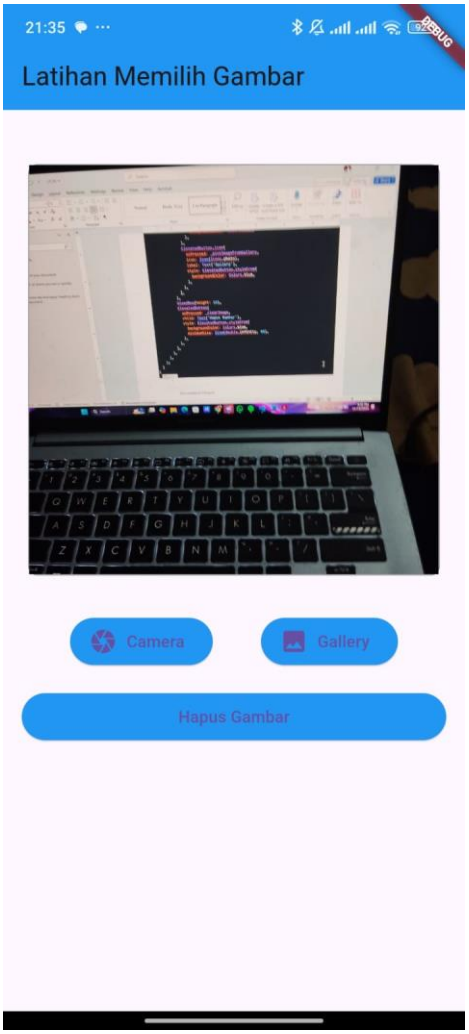
Screenshoot Output



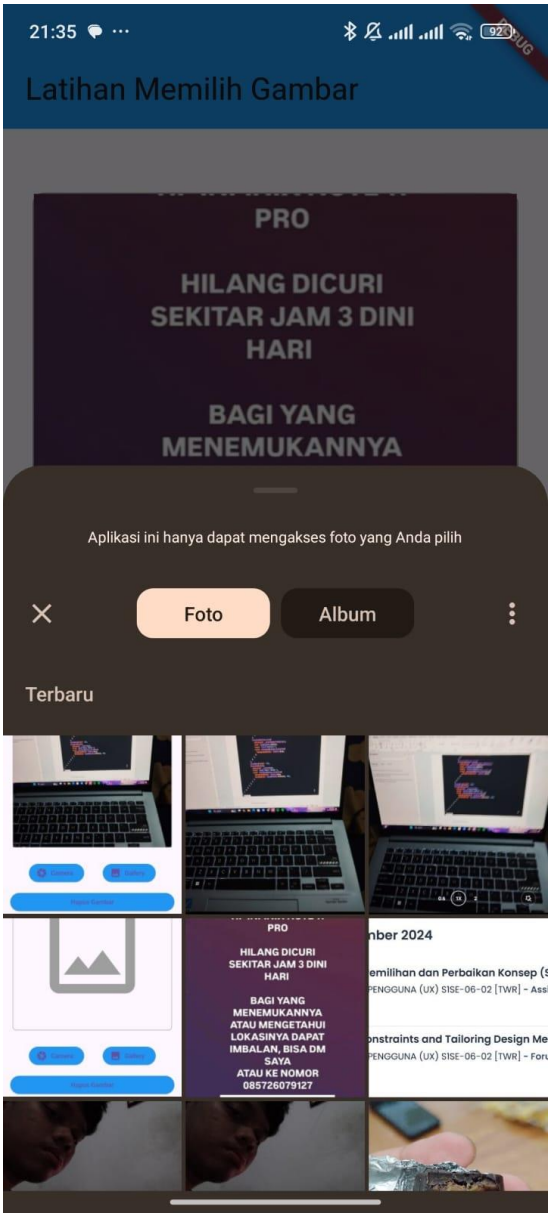
Tampilan awal aplikasi



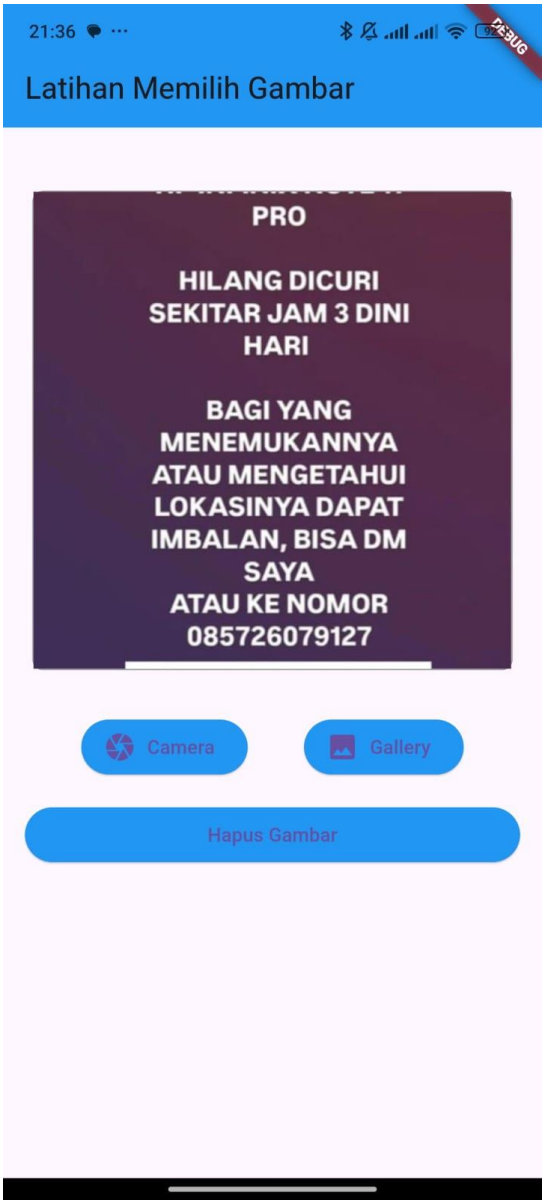
Akses Kamera



Gambar dari Kamera



Akses ke galeri perangkat



Gambar dari galeri perangkat

Deskripsi Program

Program ini adalah aplikasi Flutter yang memungkinkan pengguna memilih gambar dari galeri atau kamera menggunakan package `image_picker`. Aplikasi ini terdiri dari widget utama `MyApp` yang menampilkan halaman `ImagePickerPage`, di mana pengguna dapat melihat gambar yang dipilih dalam sebuah kotak pratinjau. Gambar dapat diambil dari kamera atau galeri dengan menekan tombol yang sesuai, dan tombol tambahan disediakan untuk menghapus gambar yang dipilih. File gambar yang dipilih disimpan dalam variabel `_imageFile`, dan ditampilkan menggunakan widget `Image.file` jika ada gambar yang tersedia.