LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL IX

API PERANGKAT KERAS

Disusun Oleh :

Dhiya Ulhaq Ramadhan 2211104053

Kelas :

SE-06-02

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

A. GUIDED

CAMERA API DAN MEDIA API

1. Main.dart

Source code :

```dart
import 'package:api/media_screen.dart';
import 'package:flutter/material.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: 'Media App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: const MediaScreen(),
    ); // MaterialApp
  }
}

```

2. camera_screen.dart

Source code :

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

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

  @override
  State<MyCameraScreen> createState() => _MyCameraScreenState();
}

class _MyCameraScreenState extends State<MyCameraScreen> {
  late CameraController _controller;
  Future<void>? _initializeControllerFuture;

  Future<void> _initializeCamera() async {
    final cameras = await availableCameras();
    final firstCamera = cameras.first;

    _controller = CameraController(
      firstCamera,
      ResolutionPreset.high,
    );

    _initializeControllerFuture = _controller.initialize();
    setState(() {});
  }

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

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Camera Implementation"),
        centerTitle: true,
        backgroundColor: Colors.greenAccent,
      ), // AppBar
      body: FutureBuilder(
        future: _initializeControllerFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.done) {
            return CameraPreview(_controller);
          } else {
            return const Center(
              child: CircularProgressIndicator(),
            ); // Center
          }
        },
      ), // FutureBuilder
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          try {
            await _initializeControllerFuture;
            final image = await _controller.takePicture();
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (_) => DisplayScreen(
                  imagePath: image.path,
                ), // DisplayScreen
              ), // MaterialPageRoute
            );
          } catch (e) {
```

```
75                print(e);
76              }
77            },
78            child: const Icon(Icons.camera),
79          ), // FloatingActionButton
80        ); // Scaffold
81      }
82    }
```

3. display_screen.dart

Source code :

```
1   import 'package:flutter/material.dart';
2   import 'dart:io';
3
4   class DisplayScreen extends StatelessWidget {
5     final String imagePath;
6
7     const DisplayScreen({
8       super.key,
9       required this.imagePath,
10    });
11
12    @override
13    Widget build(BuildContext context) {
14      return Scaffold(
15        appBar: AppBar(
16          title: const Text('Display Screen'),
17          centerTitle: true,
18          backgroundColor: Colors.greenAccent,
19          actions: [
20            IconButton(
21              icon: const Icon(Icons.share),
22              onPressed: () {
23              },
24            ), // IconButton
25          ],
26        ), // AppBar
27        body: Column(
28          children: [
29            Expanded(
30              child: Image.file(
31                File(imagePath),
32                fit: BoxFit.contain,
33              ), // Image.file
34            ), // Expanded
35            Padding(
36              padding: const EdgeInsets.all(16.0),
37              child: Row(
```

```
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                children: [
                  ElevatedButton(
                    onPressed: () {
                      Navigator.pop(context);
                    },
                    child: const Text('Take Another Photo'),
                  ), // ElevatedButton
                  ElevatedButton(
                    onPressed: () {
                    },
                    child: const Text('Save Photo'),
                  ), // ElevatedButton
                ],
              ), // Row
            ), // Padding
          ],
        ), // Column
      ); // Scaffold
    }
  }
```

4. image_picker_screen.dart
   Source Code :

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

class ImagePickerScreen extends StatefulWidget {
  final ImageSourceType type;

  ImagePickerScreen(this.type);

  @override
  ImagePickerScreenState createState() => ImagePickerScreenState(this.type);
}

class ImagePickerScreenState extends State<ImagePickerScreen> {
  File? _image;
  late ImagePicker imagePicker;
  final ImageSourceType type;

  ImagePickerScreenState(this.type);

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(type == ImageSourceType.camera
            ? "Image from Camera"
            : "Image from Gallery"), // Text
      ), // AppBar
      body: Column(
        children: <Widget>[
          SizedBox(height: 52),
```

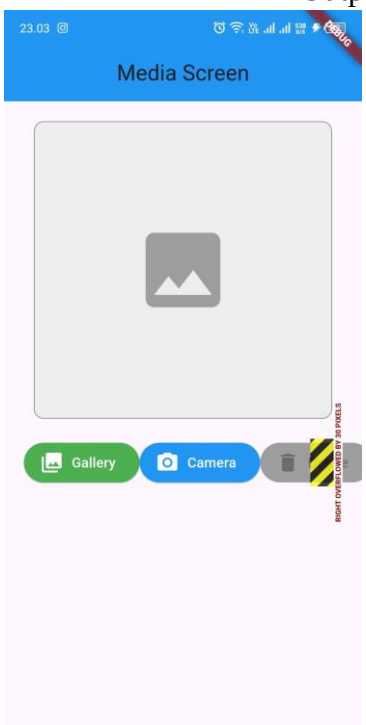
```
        Center(
          child: GestureDetector(
            onTap: () async {
              var source = type == ImageSourceType.camera
                  ? ImageSource.camera
                  : ImageSource.gallery;

              XFile? image = await imagePicker.pickImage(
                source: source,
                imageQuality: 50,
                preferredCameraDevice: CameraDevice.front,
              );

              if (image != null) {
                setState(() {
                  _image = File(image.path);
                });
              } else {
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(content: Text('No image selected!')),
                );
              }
            },
            child: Container(
              width: 200,
              height: 200,
              decoration: BoxDecoration(
                color: Colors.red[200],
              ), // BoxDecoration
              child: _image != null
                  ? Image.file(
                      _image!,
                      width: 200.0,
                      height: 200.0,
                      fit: BoxFit.fitHeight,
                    ) // Image.file
                  : Icon(
                      Icons.camera_alt,
                      color: Colors.grey[800],
                    ), // Icon
            ), // Container
          ), // GestureDetector
        ), // Center
      ], // <Widget>[]
    ), // Column
  ); // Scaffold
 }
}

enum ImageSourceType { camera, gallery }
```

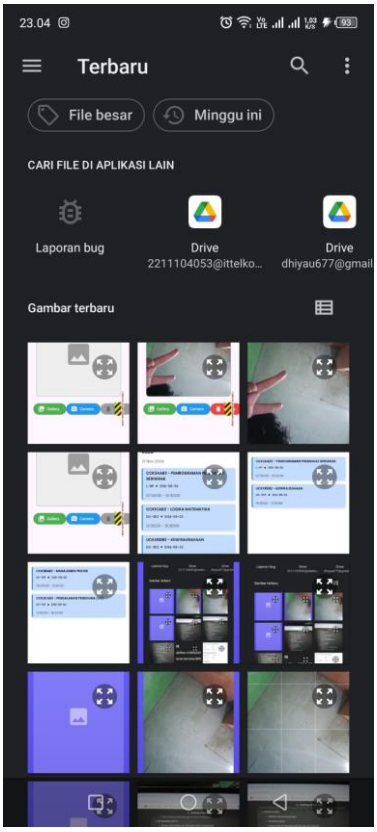
5. media_screen.dart
   Source code :

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

class MediaScreen extends StatefulWidget {
  const MediaScreen({Key? key}) : super(key: key);

  @override
  State<MediaScreen> createState() => _MediaScreenState();
}

class _MediaScreenState extends State<MediaScreen> {
  File? _selectedImage;
  final ImagePicker _imagePicker = ImagePicker();

  Future<void> _pickImage(ImageSource source) async {
    try {
      final XFile? pickedImage = await _imagePicker.pickImage(
        source: source,
        imageQuality: 80,
      );

      if (pickedImage != null) {
        setState(() {
          _selectedImage = File(pickedImage.path);
        });
      }
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Terjadi kesalahan saat mengambil gambar'),
          backgroundColor: Colors.red,
        ), // SnackBar
      );
    }
  }

  void _deleteImage() {
    setState(() {
      _selectedImage = null;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Media Screen'),
        centerTitle: true,
        backgroundColor: Colors.blue,
      ), // AppBar
      body: Column(
        children: [
          const SizedBox(height: 20),
          // Image Container
          Center(
            child: Container(
              width: 300,
              height: 300,
              decoration: BoxDecoration(
                color: Colors.grey[200],
                borderRadius: BorderRadius.circular(10),
                border: Border.all(color: Colors.grey),
              ), // BoxDecoration
              child: _selectedImage != null
                  ? ClipRRect(
                      borderRadius: BorderRadius.circular(10),
                      child: Image.file(
                        _selectedImage!,
                        fit: BoxFit.cover,
                      ), // Image.file
                    ) // ClipRRect
                  : const Icon(
                      Icons.image,
```

```
                    size: 100,
                    color: Colors.grey,
                  ), // Icon
                ), // Container
              ), // Center
              const SizedBox(height: 20),
              // Action Buttons
              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 20),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: [
                    ElevatedButton.icon(
                      onPressed: () => _pickImage(ImageSource.gallery),
                      icon: const Icon(Icons.photo_library),
                      label: const Text('Gallery'),
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.green,
                        foregroundColor: Colors.white,
                      ),
                    ), // ElevatedButton.icon
                    ElevatedButton.icon(
                      onPressed: () => _pickImage(ImageSource.camera),
                      icon: const Icon(Icons.camera_alt),
                      label: const Text('Camera'),
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.blue,
                        foregroundColor: Colors.white,
                      ),
                    ), // ElevatedButton.icon
                    ElevatedButton.icon(
                      onPressed: _selectedImage != null ? _deleteImage : null,
                      icon: const Icon(Icons.delete),
                      label: const Text('Delete'),
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.red,
                        foregroundColor: Colors.white,
                        disabledBackgroundColor: Colors.grey,
                      ),
                    ), // ElevatedButton.icon
                  ],
                ), // Row
              ), // Padding
            ],
          ), // Column
        ); // Scaffold
    }
}
```

Output :





Penjelasan Program :
Ketika MediaScreen dijalankan, layar akan menampilkan sebuah Scaffold dengan AppBar berjudul "Media Screen" di bagian atas. Di bawah AppBar, terdapat container berukuran 300x300 pixel yang awalnya kosong dan hanya menampilkan ikon gambar sebagai placeholder. Container ini memiliki latar belakang abu-abu muda dengan border melengkung dan garis tepi.Di bawah container terdapat tiga tombol: Gallery, Camera, dan Delete. Tombol Delete awalnya tidak aktif (disabled) karena belum ada gambar yang dipilih.

Saat pengguna menekan tombol Gallery, fungsi _pickImage() akan dipanggil dengan parameter ImageSource.gallery. Fungsi ini menggunakan ImagePicker untuk membuka galeri perangkat. Pengguna dapat memilih gambar dari galeri, dan setelah gambar dipilih, fungsi akan mengubah state _selectedImage dengan file gambar yang dipilih. Container kemudian akan memperbarui tampilannya untuk menampilkan gambar yang dipilih dengan BoxFit.cover agar gambar mengisi container secara proporsional.

Jika pengguna menekan tombol Camera, proses yang sama terjadi tetapi dengan ImageSource.camera sebagai parameter. Kamera perangkat akan dibuka, dan pengguna dapat mengambil foto. Setelah foto diambil, gambar akan ditampilkan dalam container yang sama.

Ketika ada gambar yang ditampilkan, tombol Delete menjadi aktif. Jika tombol Delete ditekan, fungsi _deleteImage() akan dipanggil, yang mengatur _selectedImage menjadi null, menghapus gambar dari container dan mengembalikan tampilan ke ikon placeholder.

Jika terjadi kesalahan selama proses pengambilan gambar, baik dari galeri maupun kamera, program akan menampilkan SnackBar dengan pesan error berwarna merah di bagian bawah layar. Setiap perubahan gambar (pemilihan atau penghapusan) menggunakan setState() untuk memicu rebuild widget dan memperbarui tampilan UI. Gambar ditampilkan menggunakan Image.file yang membaca file gambar langsung dari path yang tersimpan di _selectedImage.

Program menggunakan manajemen state sederhana dengan StatefulWidget karena hanya perlu melacak satu variabel state (_selectedImage). Setiap interaksi pengguna dengan tombol akan mempengaruhi state ini, yang kemudian mempengaruhi tampilan container gambar dan status tombol Delete.

B. UNGUIDED

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.

Source code :
Lib/main.dart

```dart
import 'package:flutter/material.dart';
import 'screens/image_picker_screen.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: 'Image Picker Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: const ImagePickerScreen(),
    ); // MaterialApp
  }
}

```

lib/screens/image_picker_screen.dart

```dart
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';
import '../widgets/custom_button.dart';
import '../services/image_picker_service.dart';

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

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

class _ImagePickerScreenState extends State<ImagePickerScreen> {
  File? _image;
  final ImagePickerService _imagePickerService = ImagePickerService();

  Future<void> _pickImage(ImageSource source) async {
    final pickedImage = await _imagePickerService.pickImage(source);
    if (pickedImage != null) {
      setState(() {
        _image = File(pickedImage.path);
      });
    }
  }

  void _removeImage() {
    setState(() {
      _image = null;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: const BoxDecoration(
```

```
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [Color(0xFF6C63FF), Color(0xFF5046E4)],
          ), // LinearGradient
        ), // BoxDecoration
        child: SafeArea(
          child: Padding(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                const Text(
                  'Select Image',
                  style: TextStyle(
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                    color: Colors.white,
                  ), // TextStyle
                  textAlign: TextAlign.center,
                ), // Text
                const SizedBox(height: 8),
                const Text(
                  'Choose your image from camera or gallery',
                  style: TextStyle(
                    color: Colors.white70,
                    fontSize: 16,
                  ), // TextStyle
                  textAlign: TextAlign.center,
                ), // Text
                const SizedBox(height: 40),
                Expanded(
                  child: Container(
                    decoration: BoxDecoration(
                      color: Colors.white.withOpacity(0.2),
                      borderRadius: BorderRadius.circular(20),
                    ), // BoxDecoration
```

```
                child: _image != null
                    ? ClipRRect(
                        borderRadius: BorderRadius.circular(20),
                        child: Image.file(
                          _image!,
                          fit: BoxFit.cover,
                        ), // Image.file
                      ) // ClipRRect
                    : Icon(
                        Icons.image,
                        size: 80,
                        color: Colors.white.withOpacity(0.5),
                      ), // Icon
              ), // Container
            ), // Expanded
            const SizedBox(height: 20),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                CustomButton(
                  onPressed: () => _pickImage(ImageSource.camera),
                  icon: Icons.camera_alt,
                  label: 'Camera',
                  backgroundColor: Colors.blue.shade600,
                ), // CustomButton
                CustomButton(
                  onPressed: () => _pickImage(ImageSource.gallery),
                  icon: Icons.photo_library,
                  label: 'Gallery',
                  backgroundColor: Colors.blue.shade600,
                ), // CustomButton
              ],
            ), // Row
            const SizedBox(height: 12),
            CustomButton(
              onPressed: _removeImage,
              icon: Icons.delete.
```

```
                label: 'Remove Image',
                backgroundColor: Colors.red.shade400,
                isFullWidth: true,
              ), // CustomButton
            ],
          ), // Column
        ), // Padding
      ), // SafeArea
    ), // Container
  ); // Scaffold
  }
}
```

lib/service/Image_picker_service.dart

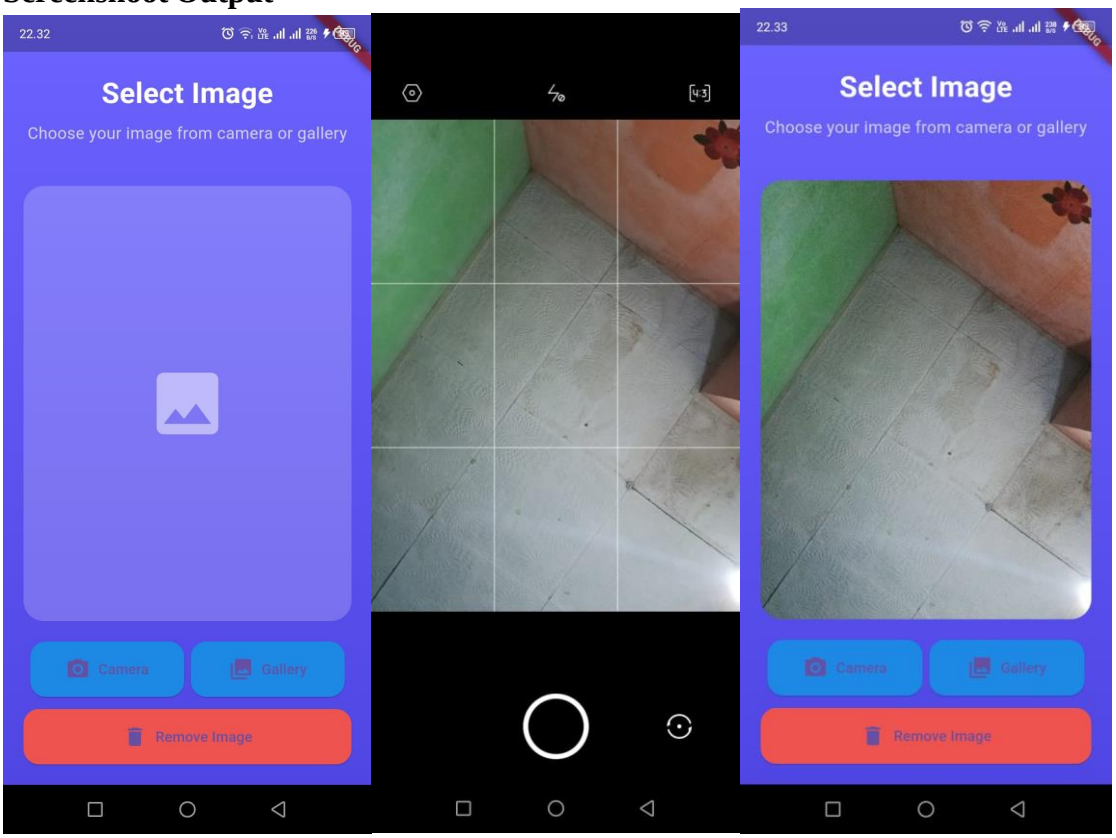
```dart
import 'package:image_picker/image_picker.dart';

class ImagePickerService {
  final ImagePicker _picker = ImagePicker();

  Future<XFile?> pickImage(ImageSource source) async {
    try {
      final XFile? image = await _picker.pickImage(
        source: source,
        imageQuality: 80,
      );
      return image;
    } catch (e) {
      print('Error picking image: $e');
      return null;
    }
  }
}

```

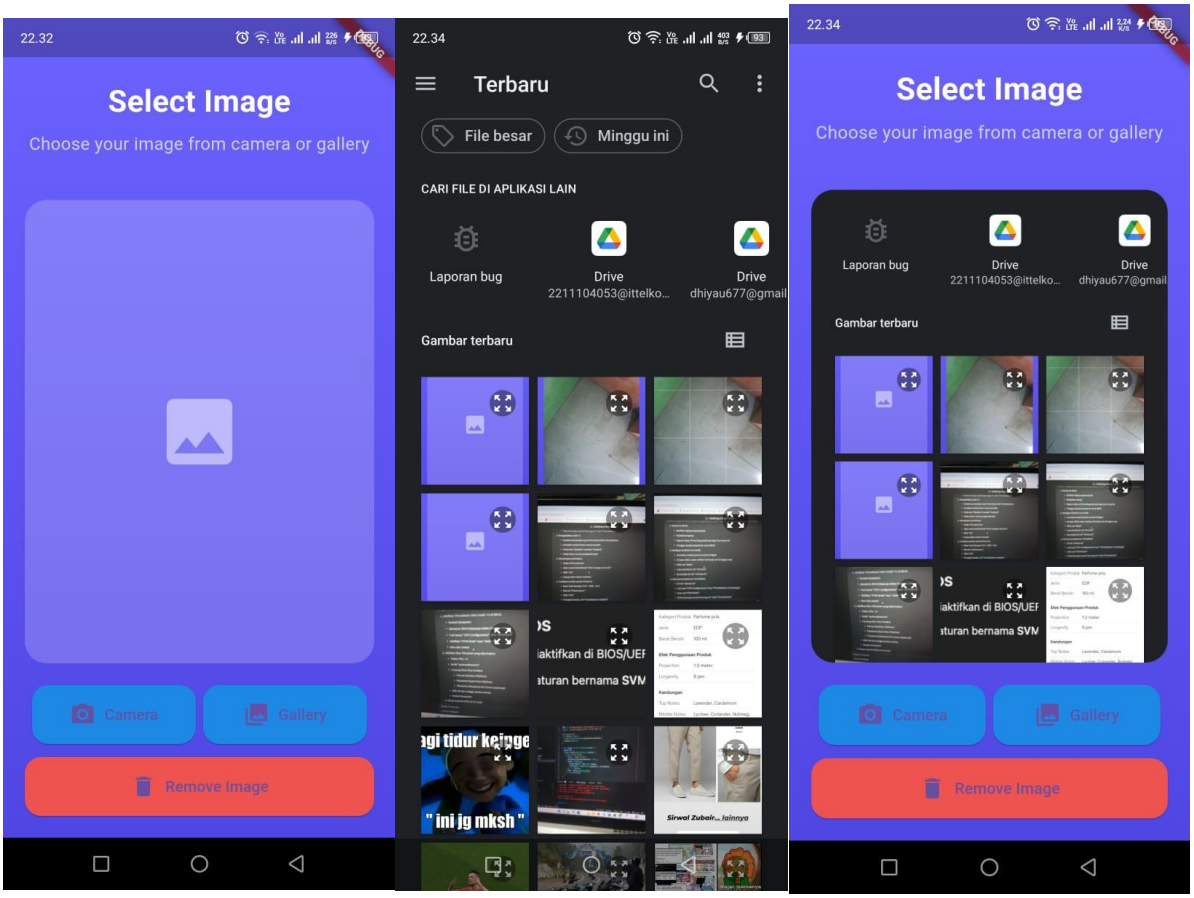
lib/widgets/custom_button.dart

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

class CustomButton extends StatelessWidget {
  final VoidCallback onPressed;
  final IconData icon;
  final String label;
  final Color backgroundColor;
  final bool isFullWidth;

  const CustomButton({
    super.key,
    required this.onPressed,
    required this.icon,
    required this.label,
    required this.backgroundColor,
    this.isFullWidth = false,
  });

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      width: isFullWidth ? double.infinity : 150,
      child: ElevatedButton(
        onPressed: onPressed,
        style: ElevatedButton.styleFrom(
          backgroundColor: backgroundColor,
          padding: const EdgeInsets.symmetric(
            horizontal: 20,
            vertical: 15,
          ), // EdgeInsets.symmetric
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(15),
          ), // RoundedRectangleBorder
        ),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
```

```
38                Icon(icon),
39                const SizedBox(width: 8),
40                Text(label),
41              ],
42            ), // Row
43          ), // ElevatedButton
44        ); // SizedBox
45      }
46    }
```

Screenshoot Output





Penjelasan Program :
Program dimulai dari file main.dart yang merupakan entry point aplikasi Flutter. Di sini, aplikasi diinisialisasi dengan MaterialApp dan menentukan ImagePickerScreen sebagai layar utama. Tema dasar aplikasi menggunakan warna biru sebagai primarySwatch.

Komponen utama aplikasi berada di ImagePickerScreen yang menampilkan antarmuka untuk memilih gambar. Layar ini memiliki latar belakang gradient dari warna ungu (#6C63FF) ke ungu yang lebih gelap (#5046E4). Di bagian atas terdapat judul "Select Image" dan subtitle "Choose your image from camera or gallery" dengan teks berwarna putih.

Bagian tengah layar menampilkan area preview gambar berbentuk container dengan latar semi-transparan. Jika belum ada gambar yang dipilih, akan ditampilkan ikon gambar default. Ketika gambar sudah dipilih, gambar tersebut akan ditampilkan dalam container dengan border radius melengkung.

Di bagian bawah layar terdapat dua tombol utama yang dibuat menggunakan CustomButton: tombol "Camera" untuk mengambil foto menggunakan kamera perangkat, dan tombol "Gallery" untuk memilih gambar dari galeri. Kedua tombol ini memiliki warna latar biru. Di bawahnya terdapat tombol "Remove Image" berwarna merah yang akan menghapus gambar yang sedang ditampilkan. Logika pemilihan gambar dikelola oleh ImagePickerService yang menggunakan package image_picker. Service ini menyediakan fungsi pickImage yang menerima parameter ImageSource (camera atau gallery) dan mengatur kualitas gambar yang dipilih menjadi 80%.

CustomButton yang digunakan untuk semua tombol dirancang dengan tampilan yang

konsisten, memiliki ikon dan label, serta dapat dikustomisasi warnanya. Tombol ini juga mendukung opsi fullWidth yang digunakan khusus untuk tombol Remove Image. Ketika pengguna memilih gambar (baik dari kamera atau galeri), gambar akan ditampilkan di area preview. Gambar dapat dihapus menggunakan tombol Remove Image, yang akan mengembalikan tampilan ke ikon default. Semua perubahan status gambar dikelola menggunakan setState untuk memastikan UI diperbarui dengan benar.

Program ini mengimplementasikan error handling sederhana di ImagePickerService untuk menangani kegagalan dalam pemilihan gambar, dengan mencetak pesan error ke console jika terjadi masalah.