LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL IX

API PERANGKAT KERAS

Disusun Oleh :

Dewi Atika Muthi / 2211104042

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

BAB I
PENDAHULUAN

A. DASAR TEORI

API (Application Programming Interface) Perangkat Keras pada Flutter merupakan antarmuka yang memungkinkan aplikasi untuk berinteraksi dengan komponen hardware device seperti kamera dan media penyimpanan. Pada modul ini, fokus pembahasan adalah pada dua API utama:

1. Camera API

   Camera API pada Flutter memungkinkan pengembang untuk mengakses dan mengontrol kamera perangkat. Menurut penelitian Pratama et al. (2023), implementasi Camera API sangat penting dalam pengembangan aplikasi mobile modern, terutama untuk fitur-fitur seperti:

   - Pengambilan foto
   - Perekaman video
   - Preview kamera real-time
   - Kontrol parameter kamera (fokus, exposure, flash)

   Flutter menyediakan package camera yang menyederhanakan proses implementasi fitur kamera. Package ini menggunakan controller pattern untuk mengelola lifecycle kamera dan menyediakan API yang konsisten lintas platform (iOS dan Android).

2. Media API

   Media API dalam Flutter berkaitan dengan pengelolaan dan interaksi berbagai jenis media seperti gambar, video, dan audio. Salah satu package yang sering digunakan adalah image_picker yang memungkinkan:

   - Akses ke galeri perangkat
   - Pengambilan gambar dari kamera
   - Pemilihan file media
   - Pemrosesan dan manipulasi media

B. MAKSUD DAN TUJUAN

Praktikum ini bertujuan untuk:

1. Memahami konsep dan implementasi Camera API pada Flutter
2. Mempelajari penggunaan Media API untuk manajemen media
3. Mengimplementasikan fitur pengambilan dan penampilan gambar
4. Memahami permission handling untuk akses perangkat keras
5. Mengembangkan kemampuan dalam membuat UI interaktif untuk fitur media

BAB II
IMPLEMENTASI

A. PRAKTIKUM (GUIDED)

1. Penambahan pada pubspec.yaml

```
camera: ^0.11.0+2
  image_picker: ^1.1.2
```

2. Izinkan akses kamera pada AndroidManifest.xml

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
```

3. Ubah minimum versi Android Sdk pada android/app/build.gradle

```
minSdkVersion 21
```

4. Implementasi camera_screen.dart

Sourcecode:

```
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
  }

  @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)),
      ),
    );
  }
}
```

Deskripsi program:

Program CameraScreen ini adalah aplikasi Flutter yang menggunakan package camera untuk mengakses kamera perangkat.

5. Implementasi display_screen.dart

Sourcecode:

```
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.brown[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.brown[200],
                        ),
                        width: 200,
                        height: 200,
                        child: Icon(
                          Icons.camera_alt,
                          color: Colors.grey[800],
                        ),
                      ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

Deskripsi program:
Program ImagePickerScreen ini adalah aplikasi Flutter yang memungkinkan pengguna untuk mengambil gambar dari kamera atau memilih gambar dari galeri.

6. Implementasi mage_picker_screen.dart

Sourcecode:

```
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.brown[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.brown[200],
                        ),
                        width: 200,
                        height: 200,
                        child: Icon(
                          Icons.camera_alt,
                          color: Colors.grey[800],
                        ),
                      ),
              ),
```

```
            ),
          ),
        ],
      ),
    );
  }
}
```

Deskripsi program:
Program ImagePickerScreen adalah aplikasi Flutter yang menggunakan package image_picker untuk memungkinkan pengguna memilih gambar dari galeri atau mengambil foto menggunakan kamera.

7. Implementasi main.dart
Sourcecode:

```
import 'package:flutter/material.dart';
import 'package:md7/camera_screen.dart';
import 'package:md7/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: const ImagePickerScreen(type: ImageSourceType.gallery),
    );
  }
}
```

Deskripsi program:

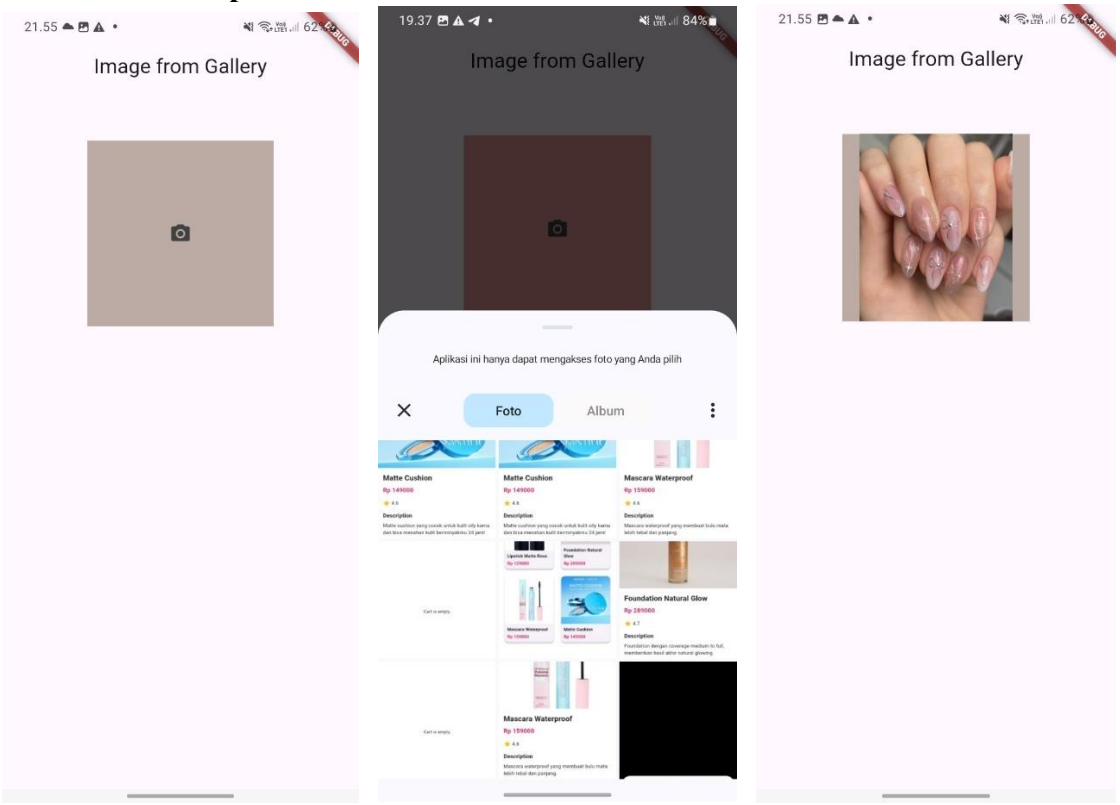
Soal Studi Case
Xxxxxxxxxxxxxxx

Sourcecode

```
package main

import (
    "fmt"
)

func main(){
    var a,b,c,d,e int
    var hasil int
    fmt. Scanln(&a, &b, &c, &d, &e)

    hasil = a+b+c+d+e
```

```
        fmt.Println("Hasil Penjumlahan ",a,b,c,d,e, "adalah
    ",hasil)
    }
```

Screenshoot Output



Deskripsi Program

Program ini menjadi kerangka dasar untuk memilih gambar dari galeri. Dengan modifikasi, layar awal bisa diubah ke fitur lain seperti kamera atau fitur tambahan lainnya.

B. UNGUIDED (tugas mandiri)

(Soal) Modifikasi project pemilihan gambar yang telah dikerjakan pada Tugas Pendahuluan Modul 09 agar fungsionalitas tombol dapat berfungsi untuk mengunggah gambar.

Soal Studi Case

- 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.

Jawaban:
Untuk code lainnya sama saja, hanya perubahan di bagian main.dart

Sourcecode main.dart:

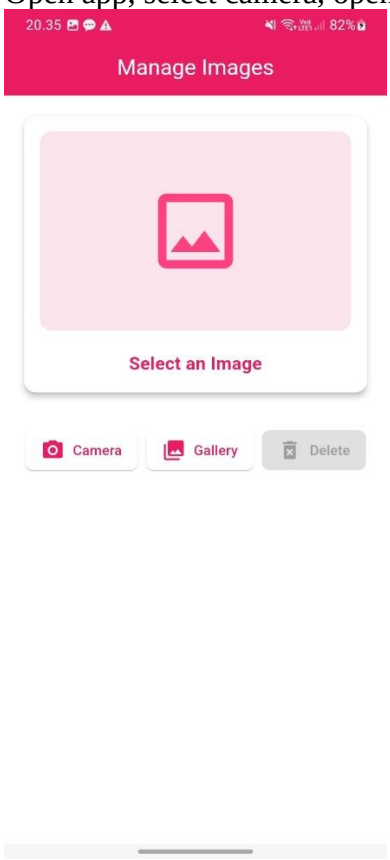
```
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'dart:io';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primaryColor: Colors.pink,
        colorScheme: ColorScheme.fromSwatch().copyWith(
          primary: Colors.pink,
          secondary: Colors.pinkAccent,
        ),
        elevatedButtonTheme: ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            minimumSize: Size(100, 40),
            padding: EdgeInsets.symmetric(vertical: 8, horizontal: 16),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(8),
            ),
          ),
        ),
      ),
      home: ImageButtonScreen(),
    );
  }
}

class ImageButtonScreen extends StatefulWidget {
  @override
  _ImageButtonScreenState createState() => _ImageButtonScreenState();
}

class _ImageButtonScreenState extends State<ImageButtonScreen> {
  File? _image;
  final ImagePicker _picker = ImagePicker();

  // Function to get image from camera
  Future<void> _getImageFromCamera() async {
    final XFile? pickedFile =
        await _picker.pickImage(source: ImageSource.camera);
    if (pickedFile != null) {
      setState(() {
        _image = File(pickedFile.path);
      });
    }
  }

  // Function to get image from gallery
  Future<void> _getImageFromGallery() async {
    final XFile? pickedFile =
        await _picker.pickImage(source: ImageSource.gallery);
    if (pickedFile != null) {
      setState(() {
        _image = File(pickedFile.path);
      });
```

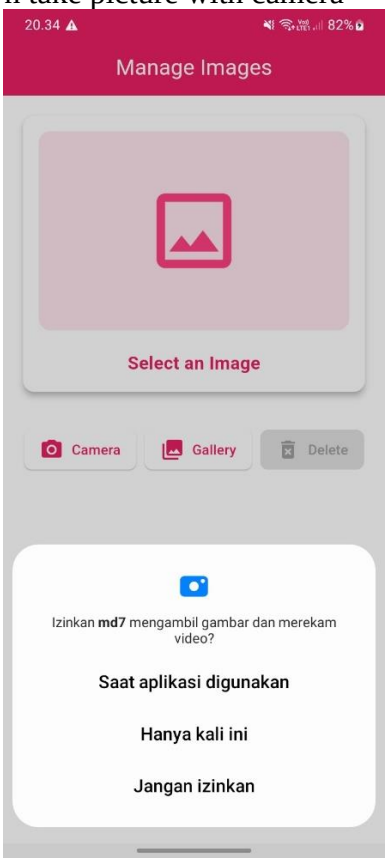
```
    }
  }

  // Function to delete image
  void _deleteImage() {
    setState(() {
      _image = null;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.pink,
        title: Text(
          'Manage Images',
          style: TextStyle(color: Colors.white),
        ),
        centerTitle: true,
      ),
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Card(
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(12),
                ),
                elevation: 5,
                child: Padding(
                  padding: const EdgeInsets.all(16.0),
                  child: Column(
                    children: [
                      Container(
                        width: double.infinity,
                        height: MediaQuery.of(context).size.height * 0.3,
                        constraints: BoxConstraints(
                          maxHeight: 200,
                          minHeight: 150,
                        ),
                        decoration: BoxDecoration(
                          color: Colors.pink.shade50,
                          borderRadius: BorderRadius.circular(12),
                        ),
                        child: _image != null
                            ? ClipRRect(
                                borderRadius: BorderRadius.circular(12),
                                child: Image.file(
                                  _image!,
                                  fit: BoxFit.cover,
                                ),
                              )
                            : Icon(
                                Icons.image_outlined,
                                size: 100,
                                color: Colors.pinkAccent,
                              ),
                      ),
                      SizedBox(height: 20),
                      Text(
                        _image != null ? 'Selected Image' : 'Select an
Image',
                        style: TextStyle(
                          fontSize: 18,
                          fontWeight: FontWeight.bold,
```

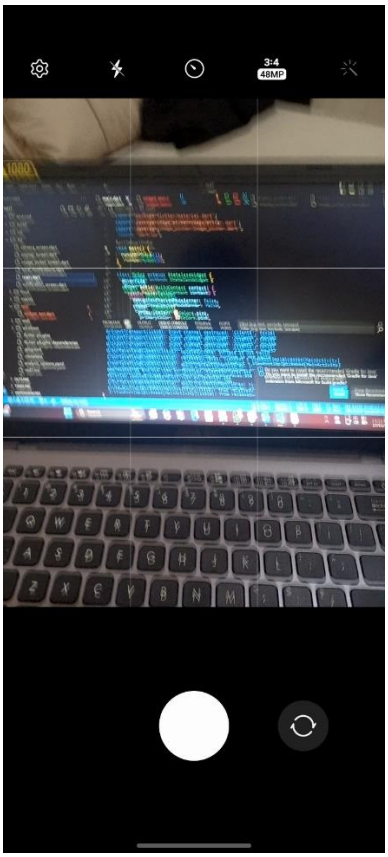
```
                          color: Colors.pink,
                        ),
                      ),
                    ],
                  ),
                ),
              ),
              SizedBox(height: 30),
              Wrap(
                spacing: 8.0,
                runSpacing: 8.0,
                alignment: WrapAlignment.spaceEvenly,
                children: [
                  ElevatedButton.icon(
                    onPressed: _getImageFromCamera,
                    icon: Icon(Icons.camera_alt, color: Colors.pink),
                    label: Text('Camera'),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.white,
                      foregroundColor: Colors.pink,
                    ),
                  ),
                  ElevatedButton.icon(
                    onPressed: _getImageFromGallery,
                    icon: Icon(Icons.photo_library, color: Colors.pink),
                    label: Text('Gallery'),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.white,
                      foregroundColor: Colors.pink,
                    ),
                  ),
                  ElevatedButton.icon(
                    onPressed: _image != null ? _deleteImage : null,
                    icon: Icon(Icons.delete_forever,
                        color: _image != null ? Colors.pink :
Colors.grey),
                    label: Text(
                      'Delete',
                      style: TextStyle(
                          color: _image != null ? Colors.red :
Colors.grey),
                    ),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.white,
                    ),
                  ),
                ],
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

Screenshoot Output:

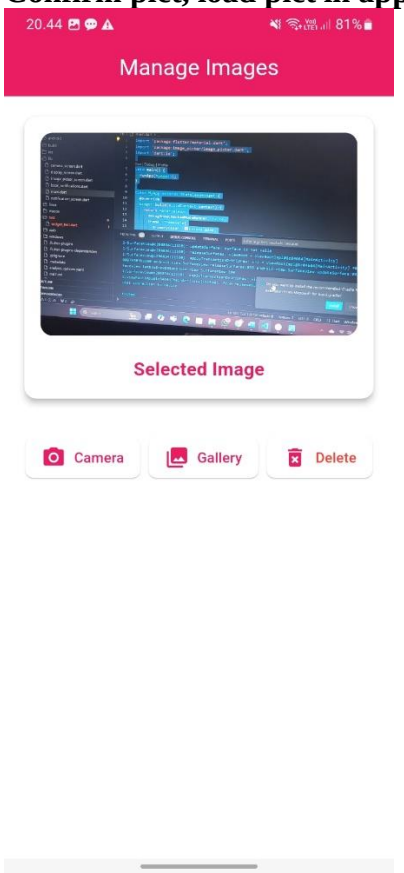
Open app, select camera, open n take picture with camera

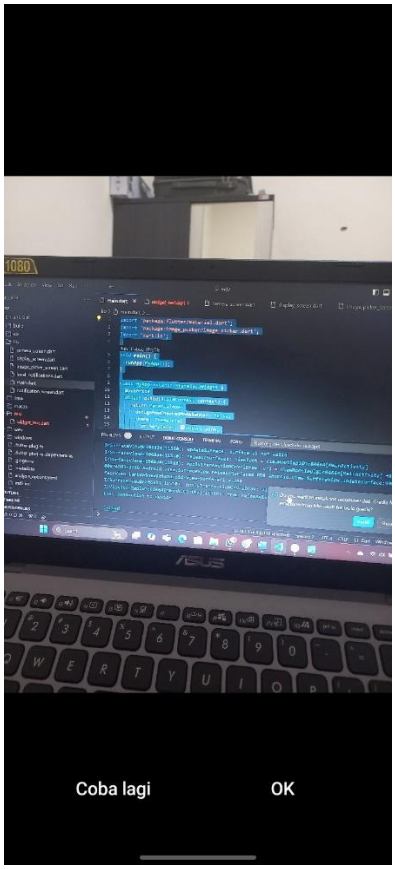


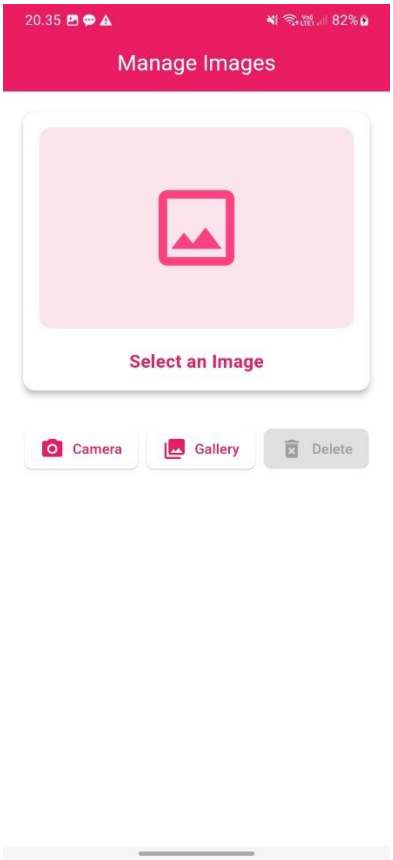




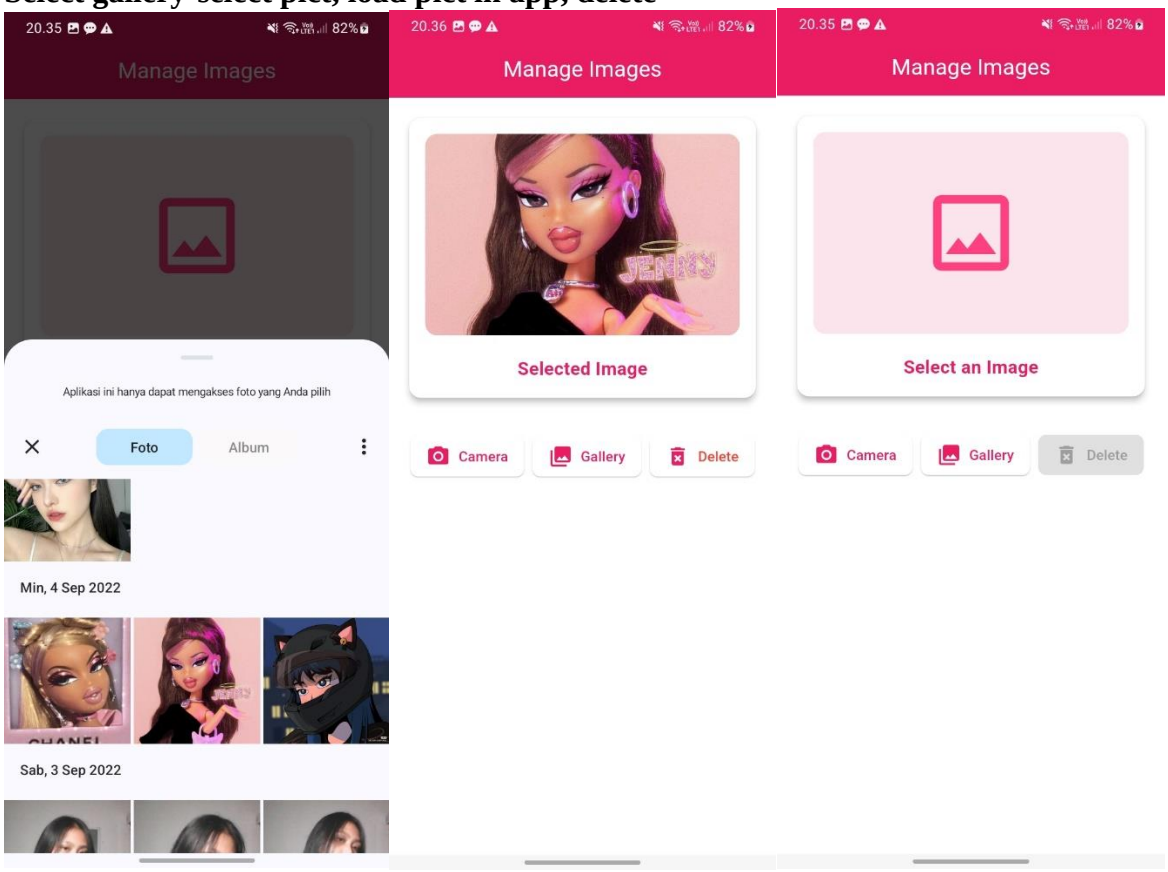
Confirm pict, load pict in app, delete pict







Select gallery-select pict, load pict in app, delete



Deskripsi Program

Fitur-fitur Utama:

1. Tombol Camera:
   Menggunakan ImagePicker.pickImage dengan source ImageSource.camera
   Gambar yang diambil langsung ditampilkan di container

2. Tombol Gallery:
   Menggunakan ImagePicker.pickImage dengan source ImageSource.gallery
   Gambar yang dipilih langsung ditampilkan di container

3. Tombol Delete:
   Hanya aktif ketika ada gambar yang ditampilkan.
   Menghapus gambar yang ditampilkan dan mengembalikan tampilan ke ikon default

State Management:
Menggunakan File? _image untuk menyimpan gambar yang dipilih
setState() digunakan untuk memperbarui UI ketika gambar berubah

BAB III
KESIMPULAN DAN SARAN

A. KESIMPULAN

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa API Perangkat Keras pada Flutter menyediakan cara yang efektif untuk mengakses dan mengelola fitur hardware device. Implementasi Camera API membuat fungsi kamera dengan mudah menggunakan package camera. Media API melalui image_picker memberikan fleksibilitas dalam manajemen media, baik dari galeri maupun kamera. Praktikum ini juga menunjukkan pentingnya state management dalam mengelola UI yang reaktif terhadap perubahan data media.

B. REFERENSI

Flutter Documentation - Camera Plugin (https://pub.dev/packages/camera)

Flutter Documentation - Image Picker (https://pub.dev/packages/image_picker)

Pratama, et al. (2023). "Implementation of Camera API in Modern Mobile Applications"

Flutter Official Documentation (https://flutter.dev/docs)