LAPORAN PRAKTIKUM
PEMROGRAMAN PERANGKAT BERGERAK

MODUL 9

API PERANGKAT KERAS

Disusun Oleh :

Dzikri Naufal Wisnu Pravida/2211104063

SE06-02

Asisten Praktikum :

Muhammad Faza Zulian

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

PRAKTIKUM

A. GUIDED

1. Camera API

Camera API berfungsi untuk memungkinkan developer (pengembang) untuk mengakses dan mengontrol kamera perangkat. Flutter menyediakan paket camera yang memudahkan implementasi fitur kamera untuk mengambil foto, merekam video, dan mengakses umpan kamera secara langsung. Paket ini sangat berguna untuk membuat aplikasi yang membutuhkan pengambilan gambar atau video, seperti aplikasi media sosial atau e-commerce.

2. Media API

Media API adalah sekumpulan alat dan pustaka yang mendukung pengelolaan dan interaksi dengan berbagai jenis media, seperti gambar, video, dan audio. Flutter tidak memiliki API media bawaan untuk semua kebutuhan media, tetapi dapat menggunakan paket-paket tambahan untuk mengakses fitur media yang umum di aplikasi.

3. Hasil Praktikum

Source Code :

main

```dart
import 'package:flutter/material.dart';
import 'package:prak_09/camera_screen.dart';

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

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyCameraScreen(),
    );
  }
}

```

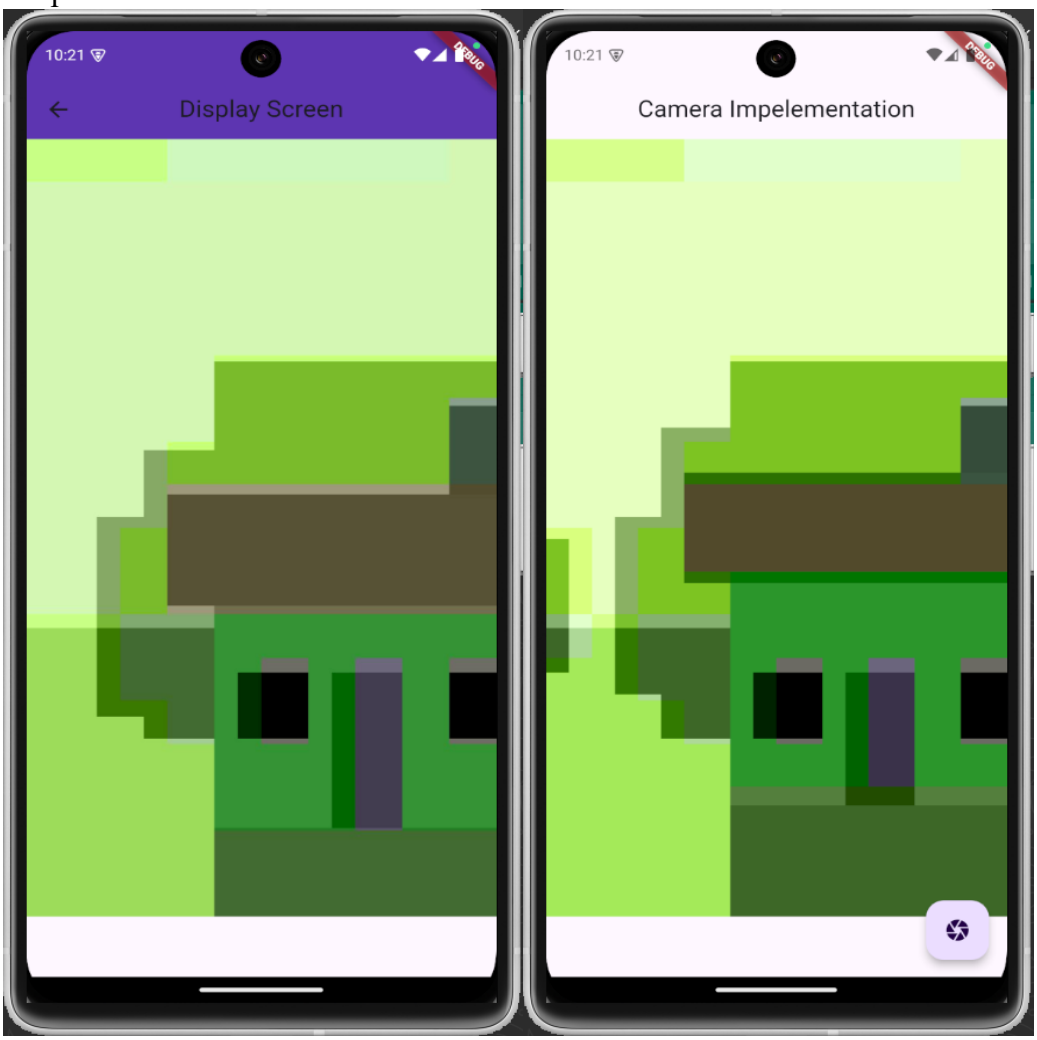
Display

```dart
import 'dart:io';

import 'package:flutter/material.dart';

class DisplayScreen extends StatelessWidget {
  final String imagePath;
  const DisplayScreen({super.key, required this.imagePath});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Display Screen"),
        centerTitle: true,
        backgroundColor: Colors.deepPurple[600],
      ),
      body: Image.file(File(imagePath)),
    );
  }
}

```

Camera

```dart
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:prak_09/display_screen.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() {
    _initializeCamera();
    super.initState();
  }

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

  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Camera Impelementation"),
        centerTitle: true,
        actions: const [],
      ),
      body: FutureBuilder(
        future: _initializeControllerFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.done) {
            return CameraPreview(_controller);
          } else {
            return Center(child: CircularProgressIndicator());
          }
        },
      ),
      floatingActionButton: FloatingActionButton(
        child: const Icon(Icons.camera),
        onPressed: () async {
          try {
            await _initializeControllerFuture;
            final image = await _controller.takePicture();
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (_) => DisplayScreen(imagePath: image.path)));
          } catch (e) {
            print(e);
          }
        },
      ),
    );
  }
}

```

Output :



B. UNGUIDED

1. Soal Studi Case

(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

Instalasi package ke dalam pubspec.yaml

dependencies:

image_picker: ^1.0.4

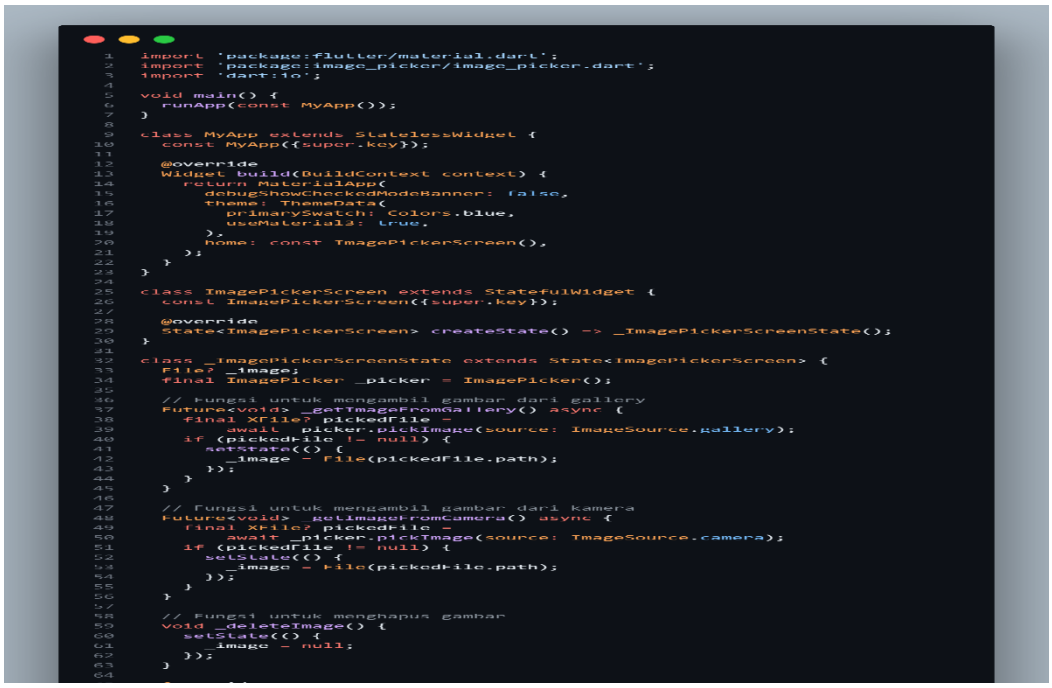
Konfigurasi platform :

Untuk Android: Tambahkan permissions di android/app/src/main/AndroidManifest.xml:

<uses-permission android:name="android.permission.CAMERA" />

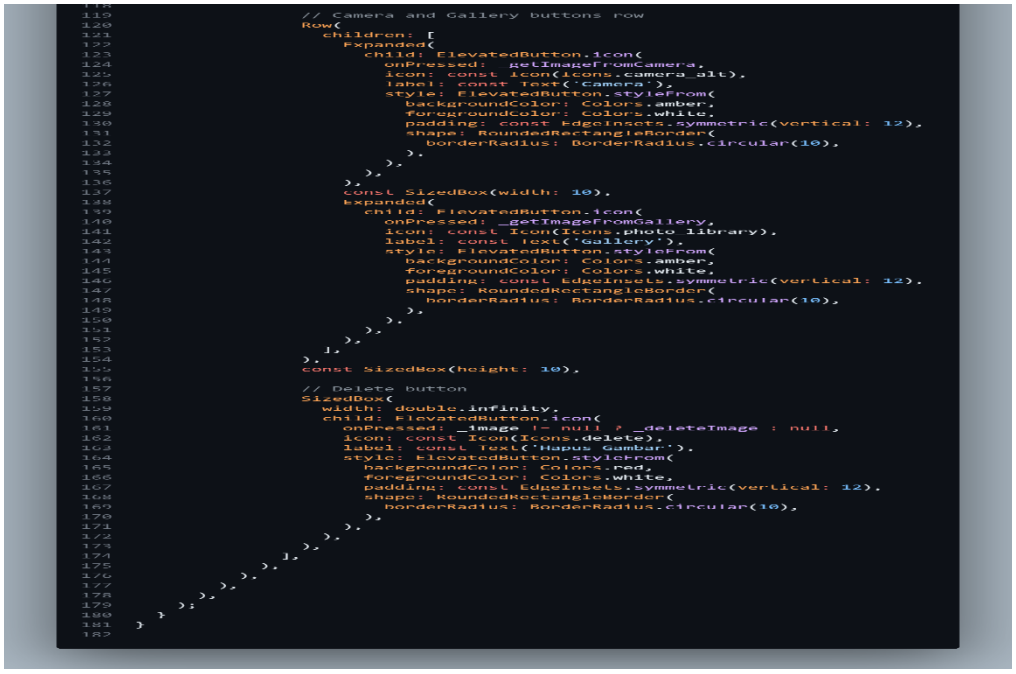
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

Modifikasi pada main.dart



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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        useMaterial3: true,
      ),
      home: const ImagePickerScreen(),
    );
  }
}

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

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

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

  // Fungsi untuk mengambil gambar dari gallery
  Future<void> _getImageFromGallery() async {
    final XFile? pickedFile =
        await _picker.pickImage(source: ImageSource.gallery);
    if (pickedFile != null) {
      setState(() {
        _image = File(pickedFile.path);
      });
    }
  }

  // Fungsi untuk mengambil gambar dari kamera
  Future<void> _getImageFromCamera() async {
    final XFile? pickedFile =
        await _picker.pickImage(source: ImageSource.camera);
    if (pickedFile != null) {
      setState(() {
        _image = File(pickedFile.path);
      });
    }
  }

  // Fungsi untuk menghapus gambar
  void _deleteImage() {
    setState(() {
      _image = null;
    });
  }
```

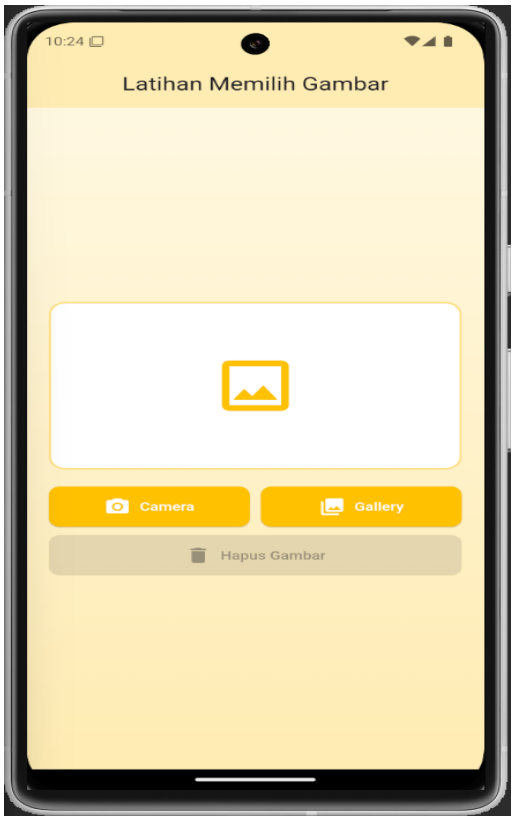
```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Latihan Memilih Gambar'),
        centerTitle: true,
        backgroundColor: Colors.amber[100],
      ),
      body: Container(
        decoration: BoxDecoration(
          gradient: LinearGradient(
            begin: Alignment.topCenter,
            end: Alignment.bottomCenter,
            colors: [
              Colors.amber.shade50,
              Colors.amber.shade100,
            ],
          ),
        ),
        child: Center(
          child: Padding(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                // Image container
                Container(
                  width: double.infinity,
                  height: 200,
                  decoration: BoxDecoration(
                    color: Colors.white,
                    borderRadius: BorderRadius.circular(15),
                    border: Border.all(
                      color: Colors.amber.shade200,
                      width: 2,
                      style: BorderStyle.solid,
                    ),
                  ),
                  child: _image != null
                      ? ClipRRect(
                          borderRadius: BorderRadius.circular(13),
                          child: Image.file(
                            _image!,
                            fit: BoxFit.cover,
                          ),
                        )
                      : const Icon(
                          Icons.image_outlined,
                          size: 80,
                          color: Colors.amber,
                        ),
                ),
                const SizedBox(height: 20),
```



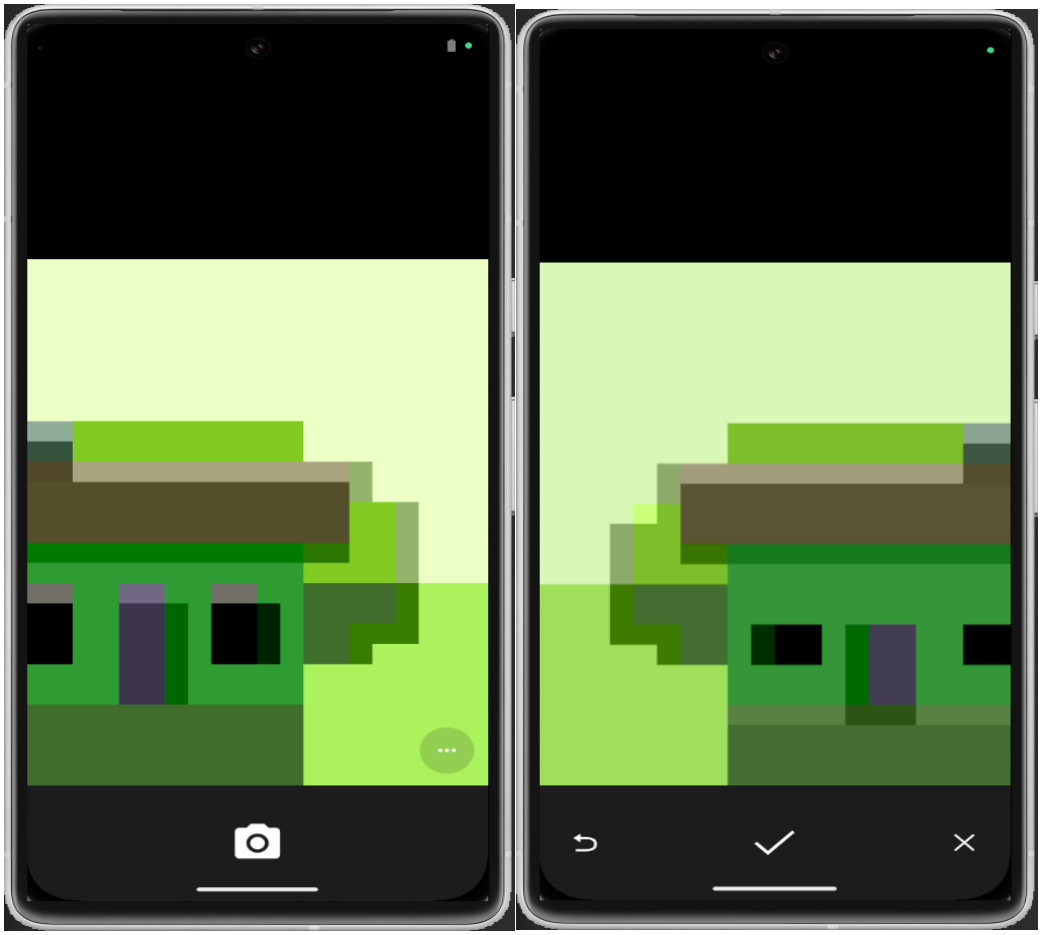
```
                // Camera and Gallery buttons row
                Row(
                  children: [
                    Expanded(
                      child: ElevatedButton.icon(
                        onPressed: _getImageFromCamera,
                        icon: const Icon(Icons.camera_alt),
                        label: const Text('Camera'),
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.amber,
                          foregroundColor: Colors.white,
                          padding: const EdgeInsets.symmetric(vertical: 12),
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(10),
                          ),
                        ),
                      ),
                    ),
                    const SizedBox(width: 10),
                    Expanded(
                      child: ElevatedButton.icon(
                        onPressed: _getImageFromGallery,
                        icon: const Icon(Icons.photo_library),
                        label: const Text('Gallery'),
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.amber,
                          foregroundColor: Colors.white,
                          padding: const EdgeInsets.symmetric(vertical: 12),
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(10),
                          ),
                        ),
                      ),
                    ),
                  ],
                ),
                const SizedBox(height: 10),

                // Delete button
                SizedBox(
                  width: double.infinity,
                  child: ElevatedButton.icon(
                    onPressed: _image != null ? _deleteImage : null,
                    icon: const Icon(Icons.delete),
                    label: const Text('Hapus Gambar'),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.red,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(vertical: 12),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(10),
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
      ),
    );
  }
}
```

Screenshoot Output

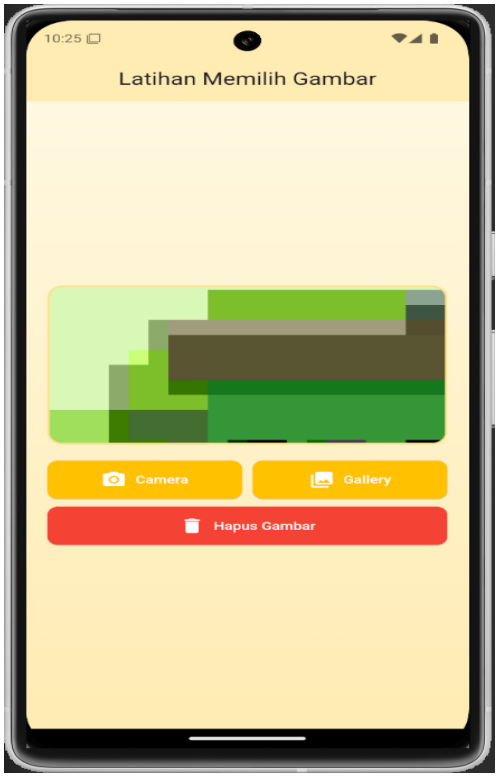
Tampilan utama



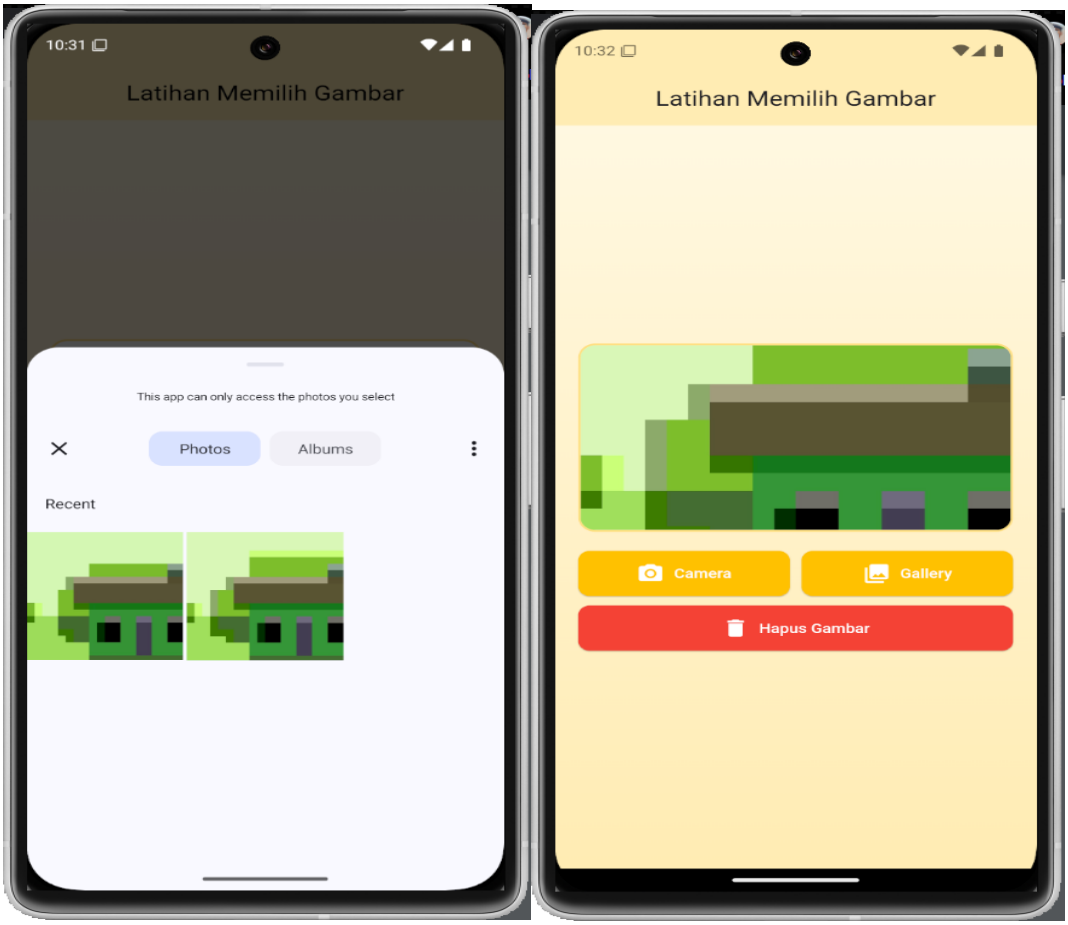
Tampilan ketika mengambil gambar dengan camera serta hasilnya



Hasilnya ketika menekan icon centang 1



Memilih ke dalam gallery



Deskripsi Program

Program ini memilih dan menampilkan gambar yang menggunakan package image_picker untuk mengakses kamera dan galeri perangkat. Aplikasi dibangun dengan StatefulWidget untuk mengelola state gambar yang dipilih, dimana status gambar disimpan dalam variabel File? _image. Program menggunakan ImagePicker untuk menangani pengambilan gambar baik dari kamera maupun galeri, yang diimplementasikan melalui dua fungsi utama _getImageFromGallery() dan _getImageFromCamera(). Kedua fungsi ini menggunakan method pickImage() dari ImagePicker dengan source yang berbeda (ImageSource.gallery atau ImageSource.camera).

Tampilan UI program terdiri dari container utama yang menampilkan gambar (jika ada) atau icon placeholder (jika tidak ada gambar), dua tombol untuk mengakses kamera dan galeri yang disusun dalam Row, serta tombol hapus gambar di bagian bawah. Container gambar menggunakan ClipRRect untuk memastikan gambar yang ditampilkan memiliki border radius yang sesuai, dan tombol hapus akan disabled (tidak aktif) ketika tidak ada gambar yang ditampilkan. Program juga menerapkan error handling sederhana dengan pengecekan null pada hasil pengambilan gambar, serta menggunakan setState untuk memperbarui UI setiap kali ada perubahan pada gambar.