LAPORAN PRAKTIKUM
Modul 09
"API PERANGKAT KERAS"

Disusun Oleh:
Ganesha Rahman Gibran -2211104058
Kelas S1SE-06-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO

2024

Tujuan

1. Mahasiswa mampu memahami konsep layout pada Flutter.
2. Mahasiswa dapat mengimplementasikan desain user interface pada Flutter.

Landasan Teori

- **Camera API dan Media API dalam Flutter**
  1. **Camera API**

     Camera API berfungsi untuk memberikan akses bagi pengembang agar dapat mengontrol kamera perangkat, memungkinkan fitur seperti pengambilan foto, perekaman video, dan umpan kamera langsung. Flutter menyediakan paket camera yang mempermudah implementasi fitur-fitur ini, terutama bagi aplikasi yang membutuhkan interaksi dengan kamera, seperti aplikasi media sosial atau e-commerce. Pada bahasan ini, kita akan menggunakan plugin camera agar aplikasi dapat mengakses kamera perangkat.

     Langkah Instalasi:

     1) Tambahkan paket camera dari Pub.dev ke dalam file pubspec.yaml.
     2) Jalankan perintah flutter pub get untuk mengunduh dependensi.
     3) Izinkan akses kamera untuk Android dengan menambahkan izin berikut ke dalam AndroidManifest.xml:

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

     4) **Konfigurasi iOS**: Pada platform iOS, plugin ini membutuhkan versi minimal iOS 10.0. Pastikan terdapat pengecekan versi dengan menggunakan plugin device_info_plus. Tambahkan juga izin berikut di file ios/Runner/Info.plist untuk menjelaskan alasan penggunaan kamera dan mikrofon:

```
  <key>NSCameraUsageDescription</key>
  <string>Perlu izin untuk mengakses kamera.</string>
  <key>NSMicrophoneUsageDescription</key>
<string>Perlu izin untuk mengakses mikrofon.</string>
```

     5) **Versi Minimum SDK**: Setel versi minimum SDK Android ke 21 atau lebih tinggi pada android/app/build.gradle:

```
minSdkVersion 21
```

     Implementasi Kamera dalam Flutter:

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

class CameraScreen extends StatefulWidget {
  @override
  _CameraScreenState createState() => _CameraScreenState();
}
```

```
class _CameraScreenState extends State<CameraScreen> {
  late CameraController _controller;
  late Future<void> _initializeControllerFuture;

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

  Future<void> _initializeCamera() async {
    final cameras = await availableCameras();
    final firstCamera = cameras.first;
    _controller = CameraController(firstCamera,
   ResolutionPreset.high);
    _initializeControllerFuture = _controller.initialize();
  }

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Contoh Kamera')),
      body: FutureBuilder<void>(
        future: _initializeControllerFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.done)
   {
            return CameraPreview(_controller);
          } else {
            return Center(child: CircularProgressIndicator());
          }
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          try {
            await _initializeControllerFuture;
            final image = await _controller.takePicture();
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
   DisplayPictureScreen(imagePath: image.path),
              ),
            );
          } catch (e) {
            print(e);
          }
```

```
        },
        child: Icon(Icons.camera_alt),
      ),
    );
  }
}

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: Text('Tampilkan Gambar')),
      body: Image.file(File(imagePath)),
    );
  }
}
```

2. Media API

Media API adalah kumpulan alat dan pustaka yang mendukung pengelolaan serta interaksi dengan berbagai media seperti gambar, video, dan audio. Meskipun Flutter tidak menyediakan API media bawaan untuk semua kebutuhan, terdapat plugin image_picker yang dapat digunakan untuk akses media dari perangkat, termasuk galeri. Plugin ini berguna bagi aplikasi yang memerlukan akses ke file media.

Konfigurasi:

Kebutuhan iOS

1) Pastikan versi iOS adalah 9.0 atau lebih tinggi. Perbarui versi minimum build pada proyek iOS jika diperlukan.
2) Tambahkan izin pada file Info.plist untuk mengakses kamera dan mikrofon:

```
<key>NSCameraUsageDescription</key>
<string>Aplikasi memerlukan akses ke kamera.</string>
<key>NSMicrophoneUsageDescription</key>
<string>Aplikasi memerlukan akses ke mikrofon.</string>
```

Implementasi mengambil gambar dari galeri atau kamera menggunakan image_picker:

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

class ImageFromGalleryEx extends StatefulWidget {
  final ImageSourceType type;
  ImageFromGalleryEx(this.type);

  @override
```

```
  ImageFromGalleryExState createState() =>
   ImageFromGalleryExState(this.type);
}

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

  ImageFromGalleryExState(this.type);

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(type == ImageSourceType.camera ? "Gambar
   dari Kamera" : "Gambar dari Galeri"),
      ),
      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);
                if (image != null) {
                  setState(() {
                    _image = File(image.path);
                  });
                }
              },
              child: Container(
                width: 200,
                height: 200,
                decoration: BoxDecoration(color:
   Colors.red[200]),
                child: _image != null
                    ? Image.file(_image!, width: 200.0, height:
   200.0, fit: BoxFit.fitHeight)
                    : Icon(Icons.camera_alt, color:
   Colors.grey[800]),
              ),
            ),
          ),
        ],
```

```
      ),
    );
  }
}

enum ImageSourceType { camera, gallery }
```

Kedua API ini meningkatkan kemampuan Flutter untuk menangani interaksi media kompleks pada platform iOS dan Android secara efektif.

Guided

1. Navigation
    - Lib/Main.dart

```
import 'package:flutter/material.dart';
import 'package:ug9/camera_screen.dart';
import 'package:ug9/image_picker_screen.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
   Colors.deepPurple),
        useMaterial3: true,
      ),
      home:
      ImageFromGalleryEx(ImageSourceType.gallery)
      // camera_screen()
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
```

```
        _counter++;
      });
    }

    @override
    Widget build(BuildContext context) {
      return Scaffold(
        appBar: AppBar(
          backgroundColor: Theme.of(context).colorScheme.inversePrimary,
          title: Text(widget.title),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              const Text(
                'You have pushed the button this many times:',
              ),
              Text(
                '$_counter',
                style: Theme.of(context).textTheme.headlineMedium,
              ),
            ],
          ),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: _incrementCounter,
          tooltip: 'Increment',
          child: const Icon(Icons.add),
        ),
      );
    }
}
```

- Lib/camera_screen.dart

```
import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:ug9/display_screen.dart';

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

  @override
  State<camera_screen> createState() => _camera_screenState();
}

class _camera_screenState extends State<camera_screen> {
  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();
  }

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

```
  }
}
```

- Lib/display_screen.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.greenAccent[600],
      ),
      body: Image.file(File(imagePath)),
    );
  }
}
```

- Lib/image_picker_screen.dart

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

enum ImageSourceType { camera, gallery }

class ImageFromGalleryEx extends StatefulWidget {
  final ImageSourceType type;

  ImageFromGalleryEx(this.type);

  @override
  ImageFromGalleryExState createState() =>
    ImageFromGalleryExState(type);
}

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

  ImageFromGalleryExState(this.type);

  @override
```

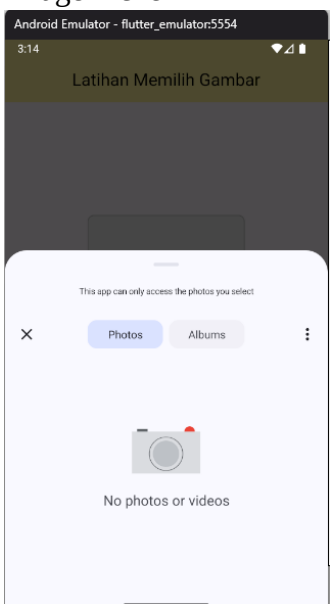
```
void initState() {
  super.initState();
  imagePicker = ImagePicker();
}

Future<void> _pickImage() 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);
    });
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(
        type == ImageSourceType.camera
        ? "Image from Camera"
        : "Image from Gallery",
      ),
    ),
    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],
              ),
              child: _image != null
                  ? Image.file(
                      _image!,
                      width: 200.0,
                      height: 200.0,
                      fit: BoxFit.fitHeight,
                    )
                  : Container(
                      width: 200,
                      height: 200,
                      decoration: BoxDecoration(
                        color: Colors.red[200],
```

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

Output :

1) Camera Screen

2) Image Picker



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.

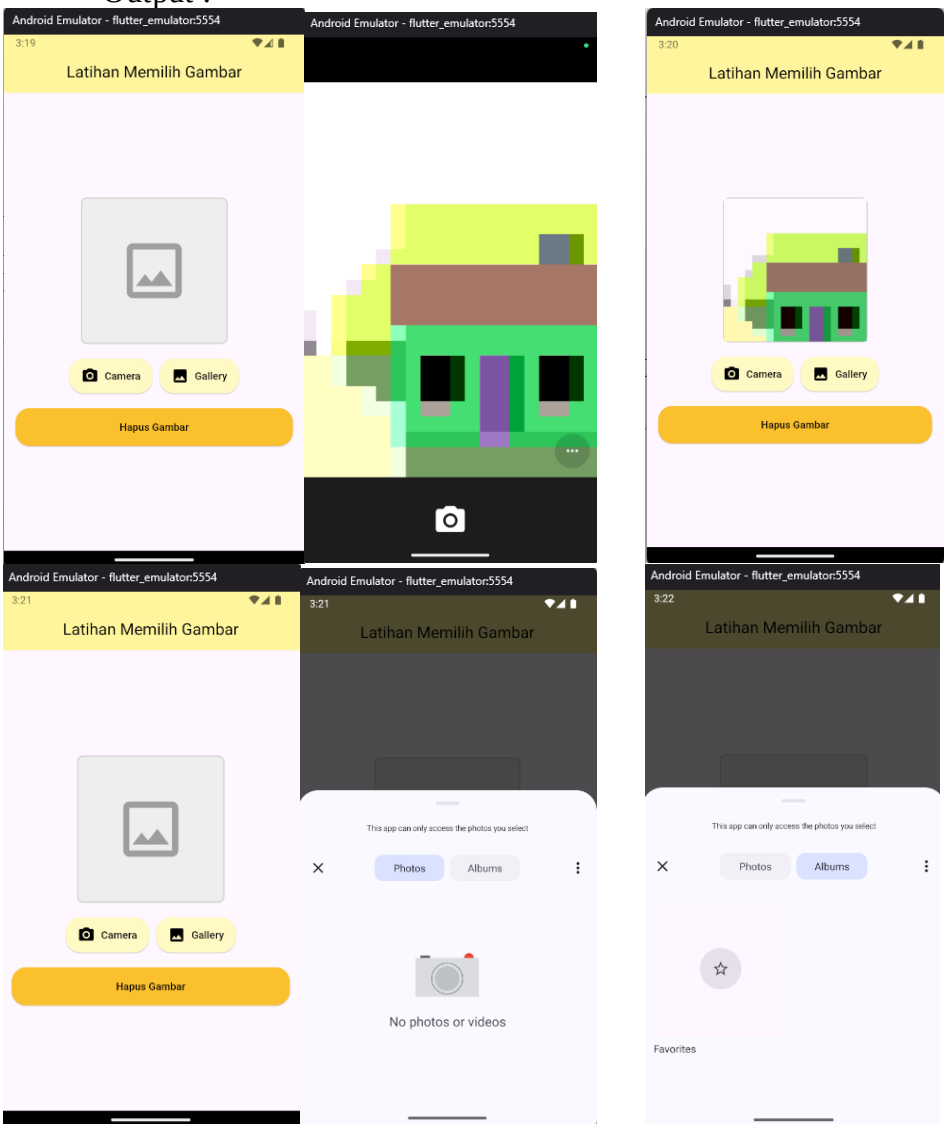
Input :

```
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(
      home: ImageSelectionScreen(),
      debugShowCheckedModeBanner: false,
    );
  }
}

class ImageSelectionScreen extends StatefulWidget {
  @override
  _ImageSelectionScreenState createState() => _ImageSelectionScreenState();
}

class _ImageSelectionScreenState extends State<ImageSelectionScreen> {
  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: Center(
          child: Text(
            'Latihan Memilih Gambar',
            style: TextStyle(color: Colors.black),
          ),
        ),
        backgroundColor: Colors.yellow.shade200,
        elevation: 0,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Container(
              width: 200,
              height: 200,
              decoration: BoxDecoration(
                color: Colors.grey.shade200,
                border: Border.all(
                  color: Colors.grey.shade400,
                  width: 1.0,
                  style: BorderStyle.solid,
                ),
                borderRadius: BorderRadius.circular(8.0),
              ),
              child: _imageFile != null
                  ? Image.file(_imageFile!, fit: BoxFit.cover)
                  : Center(
                      child: Icon(
                        Icons.image_outlined,
                        size: 100,
                        color: Colors.grey,
                      ),
                    ),
            ),
```

```
              SizedBox(height: 20),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  ElevatedButton.icon(
                    onPressed: _pickImageFromCamera,
                    icon: Icon(Icons.camera_alt, color: Colors.black),
                    label: Text('Camera', style: TextStyle(color: Colors.black)),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.yellow.shade100,
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(20),
                      ),
                      padding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),
                    ),
                  ),
                  SizedBox(width: 10),
                  ElevatedButton.icon(
                    onPressed: _pickImageFromGallery,
                    icon: Icon(Icons.photo, color: Colors.black),
                    label: Text('Gallery', style: TextStyle(color: Colors.black)),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.yellow.shade100,
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(20),
                      ),
                      padding: EdgeInsets.symmetric(horizontal: 16, vertical: 12),
                    ),
                  ),
                ],
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: _clearImage,
                child: Text('Hapus Gambar', style: TextStyle(color: Colors.black)),
                style: ElevatedButton.styleFrom(
                  backgroundColor: Colors.yellow.shade700,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(20),
                  ),
                  padding: EdgeInsets.symmetric(vertical: 16),
                  minimumSize: Size(double.infinity, 50),
                ),
              ),
            ],
          ),
        ),
      );
    }
  }
```

Output :



Deskripsi kode :

Kode di atas adalah implementasi Flutter yang memungkinkan pengguna memilih gambar dari kamera atau galeri, menampilkan gambar yang dipilih, serta menghapusnya. Menggunakan plugin image_picker, aplikasi ini menyediakan dua fungsi utama, yaitu _pickImageFromGallery untuk memilih gambar dari galeri dan _pickImageFromCamera untuk mengambil gambar dengan kamera. Gambar yang dipilih disimpan dalam variabel _imageFile dan ditampilkan di layar menggunakan widget Image.file. Jika gambar dihapus dengan tombol "Hapus Gambar," variabel ini diatur ulang menjadi null. Desain antarmuka dibuat responsif dengan tombol yang intuitif, menggunakan warna kuning lembut dan ikon yang ramah pengguna. Kode ini dapat digunakan sebagai dasar untuk aplikasi yang memerlukan pengelolaan gambar, seperti media sosial atau e-commerce.

Kesimpulan

Camera API dan **Media API** dalam pengembangan aplikasi Flutter memberikan

kemudahan bagi pengembang untuk mengintegrasikan fitur interaksi media secara efektif. **Camera API** memungkinkan kontrol penuh terhadap kamera perangkat untuk mengambil foto, merekam video, dan menyediakan umpan kamera langsung. Penggunaannya mencakup konfigurasi izin pada platform Android dan iOS serta kompatibilitas dengan berbagai versi SDK. Dengan menggunakan plugin camera, pengembang dapat dengan mudah membuat aplikasi yang mendukung fitur seperti kamera langsung atau penyimpanan gambar secara efisien. Sementara itu, **Media API** memfasilitasi pengelolaan media seperti gambar dan video yang diambil melalui kamera atau dipilih dari galeri perangkat. Dengan plugin image_picker, pengembang dapat menyediakan opsi akses media yang fleksibel, termasuk pengaturan kualitas gambar, serta memastikan kompatibilitas lintas platform. Fitur ini sangat bermanfaat untuk aplikasi yang memerlukan interaksi media, seperti mengunggah gambar atau video. Dengan kedua API tersebut, pengembang dapat menciptakan aplikasi yang lebih interaktif, fungsional, dan sesuai dengan kebutuhan pengguna, mendukung pengembangan aplikasi modern seperti media sosial, e-commerce, atau aplikasi kreatif lainnya.