TUGAS PENDAHULUAN
PEMROGRAMAN PERANGKAT BERGERAK

MODUL IX

NAVIGASI DAN NOTIFIKASI

Disusun Oleh :

Farhan Kurniawan / 2311104073

SE-07-02

Asisten Praktikum :

Yoga Eka Pratama

Zulfa Mustafa Akhyar Iswahyudi

Dosen Pengampu :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

TUGAS PENDAHULUAN

A. SOAL NOMOR 1

Source code

```
// image.picker.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(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
```

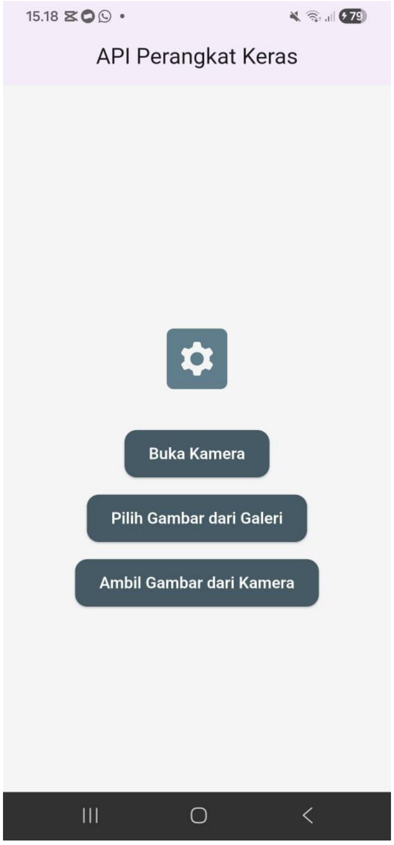
```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(
        type == ImageSourceType.camera
            ? "Image from Camera"
            : "Image from Gallery",
      ),
    ),
    body: Column(
      children: <Widget>[
        SizedBox(height: 52),
        Center(
          //mengambil gambar dari camera atau gallery
          child: GestureDetector(
            onTap: () async {
              //operasi ternary untuk memilih sumber gambar
              var source = type == ImageSourceType.camera
                  ? ImageSource.camera
                  : ImageSource.gallery;

              //menyimpan gambar pada variabel image
              XFile? image = await imagePicker.pickImage(
                source: source,
                imageQuality: 50,
                preferredCameraDevice: CameraDevice.front,
              );

              if (image != null) {
                setState(() {
                  _image = File(image.path);
                });
              }
              },
              child: Container(
                width: 200,
                height: 200,
                decoration: BoxDecoration(color: Colors.red[200]),

                // menampilkan gambar dari camera atau gallery
                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]),
                      ),
              ),
            ),
          ),
        ],
      ),
    );
  }
}

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

List<CameraDescription>? cameras;

class CameraApp extends StatefulWidget {
  @override
  _CameraAppState createState() => _CameraAppState();
}

class _CameraAppState extends State<CameraApp> {
  CameraController? controller;

  Future<void> initCamera() async {
    WidgetsFlutterBinding.ensureInitialized();
    cameras = await availableCameras();

    if (cameras != null && cameras!.isNotEmpty) {
      print(cameras);

      controller = CameraController(cameras![0], ResolutionPreset.max);
      controller?.initialize().then((_) {
        if (!mounted) {
          return;
        }
        setState(() {});
      });
    }
  }

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

  @override
  void dispose() {
    controller?.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    if (controller == null || !controller!.value.isInitialized) {
      return Container();
    }
    return MaterialApp(home: Scaffold(body: CameraPreview(controller!)));
  }
}

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

List<CameraDescription> cameras = [];

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  try {
    cameras = await availableCameras();
  } catch (e) {
    print("Error initializing cameras: $e");
  }
  runApp(const MainApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'API Perangkat Keras',
      theme: ThemeData(
        primarySwatch: Colors.blueGrey,
        scaffoldBackgroundColor: const Color.fromRGBO(245, 245, 245, 1),
        elevatedButtonTheme: ElevatedButtonThemeData(
          style: ElevatedButton.styleFrom(
            padding: const EdgeInsets.symmetric(horizontal: 24, vertical: 14),
            backgroundColor: Colors.blueGrey[700],
            foregroundColor: Colors.white,
            textStyle: const TextStyle(
              fontSize: 16,
              fontWeight: FontWeight.w600,
            ),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(12),
            ),
          ),
        ),
      ),
      home: const HomePage(),
    );
  }
```

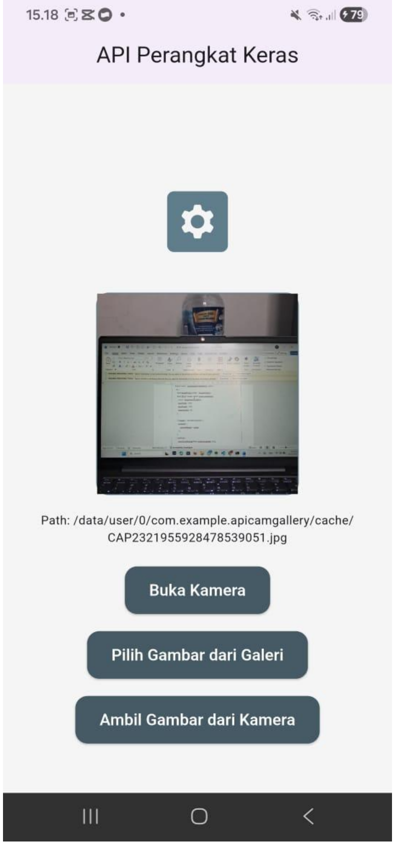
```
}

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

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  XFile? _selectedImage;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("API Perangkat Keras"),
        centerTitle: true,
        elevation: 2,
      ),
      body: Center(
        child: Padding(
          padding: const EdgeInsets.all(24.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              const Icon(
                Icons.settings_applications_rounded,
                size: 80,
                color: Colors.blueGrey,
              ),
              const SizedBox(height: 30),
// Tampilkan gambar yang dipilih jika ada
if (_selectedImage != null) ...[
  Container(
    width: 200,
    height: 200,
    decoration: BoxDecoration(
      border: Border.all(color: Colors.blueGrey),
      borderRadius: BorderRadius.circular(12),
    ),
    child: Image.file(
      File(_selectedImage!.path),
      fit: BoxFit.cover,
      errorBuilder: (context, error, stackTrace) {
        return const Center(
          child: Icon(Icons.error, color: Colors.red),
        );
      },
    ),
  ),
  const SizedBox(height: 16),
  Text(
    'Path: ${_selectedImage!.path}',
    style: const TextStyle(fontSize: 12),
    textAlign: TextAlign.center,
  ),
  const SizedBox(height: 20),
],

ElevatedButton(
  onPressed: () async {
    await _openCamera(context);
  },
  child: const Text("Buka Kamera"),
      ),
      const SizedBox(height: 16),
      ElevatedButton(
       onPressed: () async {
        await _pickImageFromGallery();
       },
       child: const Text("Pilih Gambar dari Galeri"),
      ),
      const SizedBox(height: 16),
      ElevatedButton(
       onPressed: () async {
        await _pickImageFromCamera();
       },
       child: const Text("Ambil Gambar dari Kamera"),
      ),
     ],
    ),
   ),
  ),
 );
}

// Method untuk membuka kamera lengkap
Future<void> _openCamera(BuildContext context) async {
 try {
  if (cameras.isEmpty) {
   _showErrorDialog('Tidak ada kamera yang tersedia');
   return;
  }

  final result = await Navigator.push(
   context,
   MaterialPageRoute(builder: (context) => CameraApp(cameras: cameras)),
    );

    if (result != null && result is XFile && mounted) {
      setState(() {
        _selectedImage = result;
      });
    }
  } catch (e) {
    _showErrorDialog('Error mengakses kamera: $e');
  }
}

// Method untuk memilih gambar dari galeri
Future<void> _pickImageFromGallery() async {
  try {
    final ImagePicker picker = ImagePicker();
    final XFile? image = await picker.pickImage(
      source: ImageSource.gallery,
      maxWidth: 1200,
      maxHeight: 1200,
      imageQuality: 80,
    );

    if (image != null && mounted) {
      setState(() {
        _selectedImage = image;
      });
    }
  } catch (e) {
    _showErrorDialog('Error memilih gambar: $e');
  }
}
```

```
// Method untuk mengambil gambar dari kamera
Future<void> _pickImageFromCamera() async {
  try {
    final ImagePicker picker = ImagePicker();
    final XFile? image = await picker.pickImage(
      source: ImageSource.camera,
      maxWidth: 1200,
      maxHeight: 1200,
      imageQuality: 80,
    );

    if (image != null && mounted) {
      setState(() {
        _selectedImage = image;
      });
    }
  } catch (e) {
    _showErrorDialog('Error mengambil gambar: $e');
  }
}

void _showErrorDialog(String message) {
  if (mounted) {
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: const Text('Error'),
        content: Text(message),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('OK'),
          ),
          ],
        ),
      );
    }
  }
}

// Class CameraApp untuk fitur kamera lengkap
class CameraApp extends StatefulWidget {
  final List<CameraDescription> cameras;

  const CameraApp({super.key, required this.cameras});

  @override
  State<CameraApp> createState() => _CameraAppState();
}

class _CameraAppState extends State<CameraApp> {
  CameraController? _controller;
  bool _isCameraReady = false;
  bool _isTakingPicture = false;

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

  Future<void> _initializeCamera() async {
    try {
      _controller = CameraController(
        widget.cameras.first,
        ResolutionPreset.medium,
      enableAudio: false,
    );

    await _controller!.initialize();

    if (mounted) {
      setState(() {
        _isCameraReady = true;
      });
    }
  } catch (e) {
    if (mounted) {
      _showErrorDialog('Gagal menginisialisasi kamera: $e');
    }
  }
}

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

Future<void> _takePicture() async {
  if (!_isCameraReady ||
      _controller == null ||
      !_controller!.value.isInitialized) {
    return;
  }

  if (_isTakingPicture) return;

  setState(() {
      _isTakingPicture = true;
    });

    try {
      final XFile picture = await _controller!.takePicture();

      if (mounted) {
        Navigator.pop(context, picture);
      }
    } catch (e) {
      if (mounted) {
        _showErrorDialog('Gagal mengambil gambar: $e');
      }
    } finally {
      if (mounted) {
        setState(() {
          _isTakingPicture = false;
        });
      }
    }
  }

  void _showErrorDialog(String message) {
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: const Text('Error'),
        content: Text(message),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('OK'),
          ),
        ],
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    if (!_isCameraReady || _controller == null) {
      return Scaffold(
        appBar: AppBar(title: Text('Kamera')),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              CircularProgressIndicator(),
              SizedBox(height: 16),
              Text('Menyiapkan kamera...'),
            ],
          ),
        ),
      );
    }

    return Scaffold(
      appBar: AppBar(
        title: const Text('Kamera'),
        backgroundColor: Colors.blueGrey[700],
        actions: [
          IconButton(
            icon: const Icon(Icons.switch_camera),
            onPressed: _switchCamera,
          ),
        ],
      ),
      body: Stack(
        children: [
          CameraPreview(_controller!),
          if (_isTakingPicture)
            const Center(child: CircularProgressIndicator()),
        ],
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _isTakingPicture ? null : _takePicture,
        backgroundColor: _isTakingPicture ? Colors.grey : Colors.white,
        child: Icon(
          Icons.camera,
          color: _isTakingPicture ? Colors.white : Colors.black,
        ),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
    );
  }

  Future<void> _switchCamera() async {
    if (widget.cameras.length < 2) return;

    final newCamera = _controller!.description == widget.cameras.first
        ? widget.cameras.last
        : widget.cameras.first;

    await _controller!.dispose();

    setState(() {
      _isCameraReady = false;
    });
```

```
    _controller = CameraController(
      newCamera,
      ResolutionPreset.medium,
      enableAudio: false,
    );

    try {
      await _controller!.initialize();

      if (mounted) {
        setState(() {
          _isCameraReady = true;
        });
      }
    } catch (e) {
      if (mounted) {
        _showErrorDialog('Gagal menginisialisasi kamera baru: $e');
      }
    }
  }
}
```

Screenshoot Output





Path: /data/user/0/com.example.apicamgallery/cache/CAP2321955928478539051.jpg

Deskripsi Program

Program ini dibuat untuk memanfaatkan API perangkat keras pada smartphone seperti kamera dan galeri menggunakan Flutter, di mana pengguna dapat membuka kamera, mengambil gambar, atau memilih gambar dari galeri, kemudian menampilkannya langsung pada halaman utama. Algoritma program dimulai dengan inisialisasi kamera menggunakan availableCameras(), lalu menyiapkan CameraController untuk menampilkan preview dan menangkap gambar. Ketika pengguna menekan tombol tertentu, program menjalankan proses pengambilan atau pemilihan gambar menggunakan library image_picker, kemudian menyimpan hasilnya ke variabel _selectedImage. Setelah gambar berhasil diambil atau dipilih, program menampilkannya melalui widget Image.file bersama informasi lokasi file sebagai output. Program juga menangani kemungkinan error seperti kegagalan akses kamera atau galeri, sehingga tetap memberikan pengalaman yang aman dan terkontrol bagi pengguna.