

**TUGAS PENDAHULUAN / TUGAS UNGUIDED
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL IX
API PERANGKAT KERAS**



**Disusun Oleh :
Fahmi hasan asagaf 2311104074
SE 07 02**

**Asisten Praktikum :
Zulfa Mustafa Akhyar Iswahyudi
Yoga Eka Pratama**

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

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

TUGAS UNGUIDED

A. SOAL

Tugas Mandiri (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.

Note: Jangan lupa sertakan source code, screenshot output, dan deskripsi program.

Kreatifitas menjadi nilai tambah.

B. JAWABAN

Main.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Pilih Gambar',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
          seedColor: const Color(0xFF87CEEB),
          brightness: Brightness.light,
        ),
        useMaterial3: true,
      ),
      home: const ImagePickerScreen(),
    );
  }
}

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

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

class _ImagePickerScreenState extends State<ImagePickerScreen> {
  File? _selectedImage;
  final ImagePicker _picker = ImagePicker();
  bool _isLoading = false;

  // Fungsi untuk meminta izin
  Future<bool> _requestPermission(Permission permission) async {
    final status = await permission.request();
    return status.isGranted;
  }

  // Fungsi untuk mengambil gambar dari kamera
  Future<void> _getImageFromCamera() async {
    try {

```

```

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

        // Meminta izin kamera
        final permissionGranted = await _requestPermission(Permission.camera);

        if (!permissionGranted) {
            if (!mounted) return;
            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                    content: Text('Izin akses kamera dibutuhkan'),
                    backgroundColor: Colors.red,
                ),
            );
            return;
        }

        final XFile? image = await _picker.pickImage(
            source: ImageSource.camera,
            maxWidth: 800,
            maxHeight: 800,
            imageQuality: 80,
        );

        if (image != null) {
            setState(() {
                _selectedImage = File(image.path);
            });

            if (!mounted) return;
            ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(
                    content: const Text('Gambar dari kamera berhasil diambil'),
                    backgroundColor: Colors.green[400],
                ),
            );
        }
    } catch (e) {
        if (!mounted) return;
        ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
                content: Text('Error: $e'),
                backgroundColor: Colors.red,
            ),
        );
    } finally {
        setState(() {
            _isLoading = false;
        });
    }
}

// Fungsi untuk mengambil gambar dari galeri

```

```

Future<void> _getImageFromGallery() async {
  try {
    setState(() {
      _isLoading = true;
    });

    // Meminta izin galeri (photos)
    final permissionGranted = await _requestPermission(Permission.photos);

    if (!permissionGranted) {
      if (!mounted) return;
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Izin akses galeri dibutuhkan'),
          backgroundColor: Colors.red,
        ),
      );
      return;
    }

    final XFile? image = await _picker.pickImage(
      source: ImageSource.gallery,
      maxWidth: 800,
      maxHeight: 800,
      imageQuality: 80,
    );

    if (image != null) {
      setState(() {
        _selectedImage = File(image.path);
      });

      if (!mounted) return;
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: const Text('Gambar dari galeri berhasil dipilih'),
          backgroundColor: Colors.green[400],
        ),
      );
    }
  } catch (e) {
    if (!mounted) return;
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Error: $e'),
        backgroundColor: Colors.red,
      ),
    );
  } finally {
    setState(() {
      _isLoading = false;
    });
  }
}

```

```

// Fungsi untuk menghapus gambar
void _deleteImage() {
  if (_selectedImage == null) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text('Tidak ada gambar untuk dihapus'),
        backgroundColor: Colors.orange,
      ),
    );
    return;
  }

  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Hapus Gambar'),
        content: const Text('Apakah Anda yakin ingin menghapus gambar ini?'),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: const Text('Batal'),
          ),
          TextButton(
            onPressed: () {
              setState(() {
                _selectedImage = null;
              });
              Navigator.of(context).pop();
              ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                  content: Text('Gambar berhasil dihapus'),
                  backgroundColor: Colors.green,
                ),
              );
            },
            child: const Text(
              'Hapus',
              style: TextStyle(color: Colors.red),
            ),
          ),
        ],
      );
    },
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color(0xFF5F9FF),

```

```

appBar: AppBar(
  title: const Text(
    'Pilih Gambar',
    style: TextStyle(
      fontWeight: FontWeight.w600,
      color: Colors.white,
    ),
  ),
  backgroundColor: const Color(0xFF64B5F6),
  elevation: 2,
  centerTitle: true,
),
body: _isLoading
  ? const Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        CircularProgressIndicator(
          valueColor: AlwaysStoppedAnimation<Color>(Color(0xFF64B5F6)),
        ),
        SizedBox(height: 16),
        Text(
          'Memuat gambar...',
          style: TextStyle(
            color: Color(0xFF64B5F6),
            fontSize: 16,
          ),
        ),
      ],
    ),
  )
  : Center(
    child: Container(
      margin: const EdgeInsets.all(24),
      padding: const EdgeInsets.all(32),
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(20),
        boxShadow: [
          BoxShadow(
            color: Colors.blue.withOpacity(0.1),
            blurRadius: 15,
            offset: const Offset(0, 5),
          ),
        ],
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.min,
        children: <Widget>[
          // Container untuk menampilkan gambar atau ikon default
          Container(
            width: 250,
            height: 250,
            decoration: BoxDecoration(

```

```

        color: const Color(0xFFE3F2FD),
        borderRadius: BorderRadius.circular(20),
        border: Border.all(
          color: const Color(0xFFBBDEFB),
          width: 2,
        ),
      ),
    ),
    child: _selectedImage == null
      ? const Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Icon(
              Icons.image_outlined,
              size: 80,
              color: Color(0xFF64B5F6),
            ),
            SizedBox(height: 8),
            Text(
              'Tidak ada gambar',
              style: TextStyle(
                color: Color(0xFF64B5F6),
                fontSize: 14,
              ),
            ),
          ],
        )
      : ClipRRect(
          borderRadius: BorderRadius.circular(18),
          child: Image.file(
            _selectedImage!,
            fit: BoxFit.cover,
            width: double.infinity,
            height: double.infinity,
          ),
        ),
    ),
    const SizedBox(height: 40),

    // Container untuk tombol Camera dan Gallery
    Container(
      padding: const EdgeInsets.symmetric(horizontal: 16, vertical:
12),

      decoration: BoxDecoration(
        color: const Color(0xFFFF1F8FF),
        borderRadius: BorderRadius.circular(16),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          Expanded(
            child: _buildActionButton(
              icon: Icons.camera_alt,
              label: 'Camera',
              color: const Color(0xFF4FC3F7),

```

```

        onPressed: _getImageFromCamera,
      ),
    ),
    const SizedBox(width: 16),
    Expanded(
      child: _buildActionButton(
        icon: Icons.photo_library,
        label: 'Gallery',
        color: const Color(0xFF4FC3F7),
        onPressed: _getImageFromGallery,
      ),
    ),
  ],
),
const SizedBox(height: 24),

// Tombol Hapus Gambar
SizedBox(
  width: double.infinity,
  child: _buildActionButton(
    icon: Icons.delete,
    label: 'Hapus Gambar',
    color: const Color(0xFFEF5350),
    onPressed: _deleteImage,
  ),
),
],
),
),
),
),
);
}

```

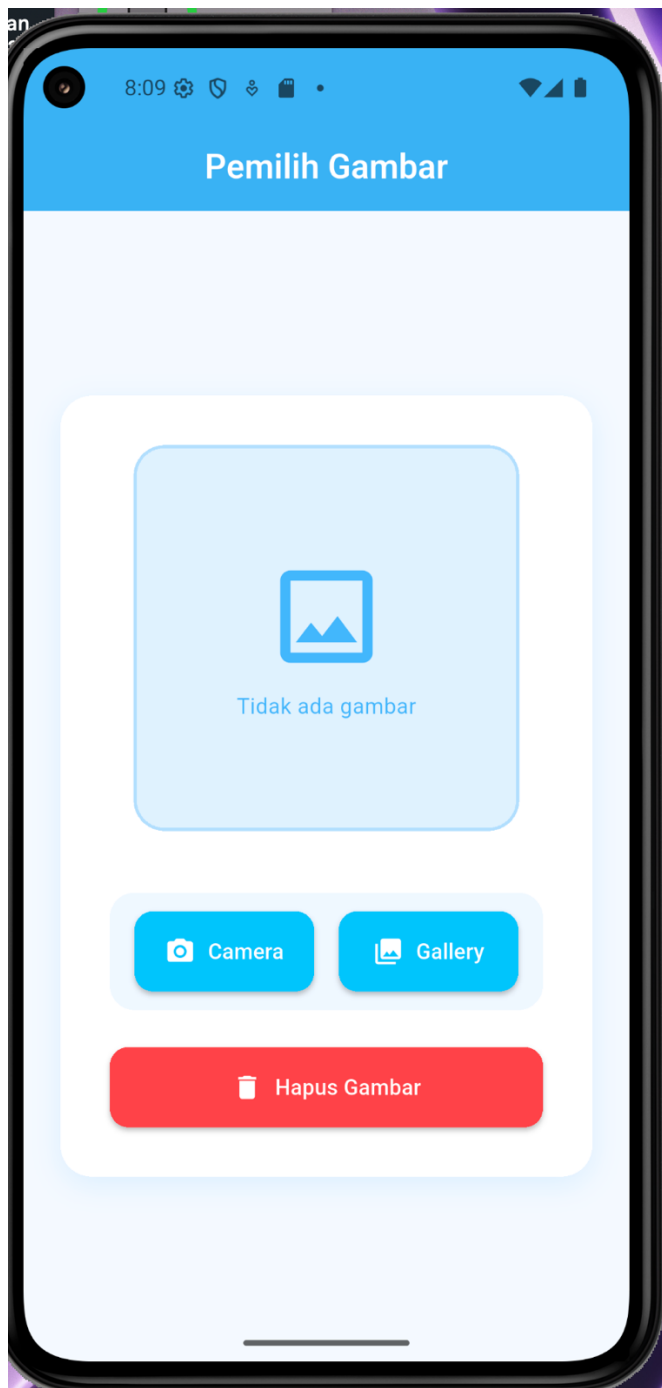
```

Widget _buildActionButton({
  required IconData icon,
  required String label,
  required Color color,
  required VoidCallback onPressed,
}) {
  return ElevatedButton.icon(
    onPressed: onPressed,
    icon: Icon(
      icon,
      size: 20,
    ),
    label: Text(
      label,
      style: const TextStyle(
        fontWeight: FontWeight.w500,
      ),
    ),
    style: ElevatedButton.styleFrom(
      backgroundColor: color,

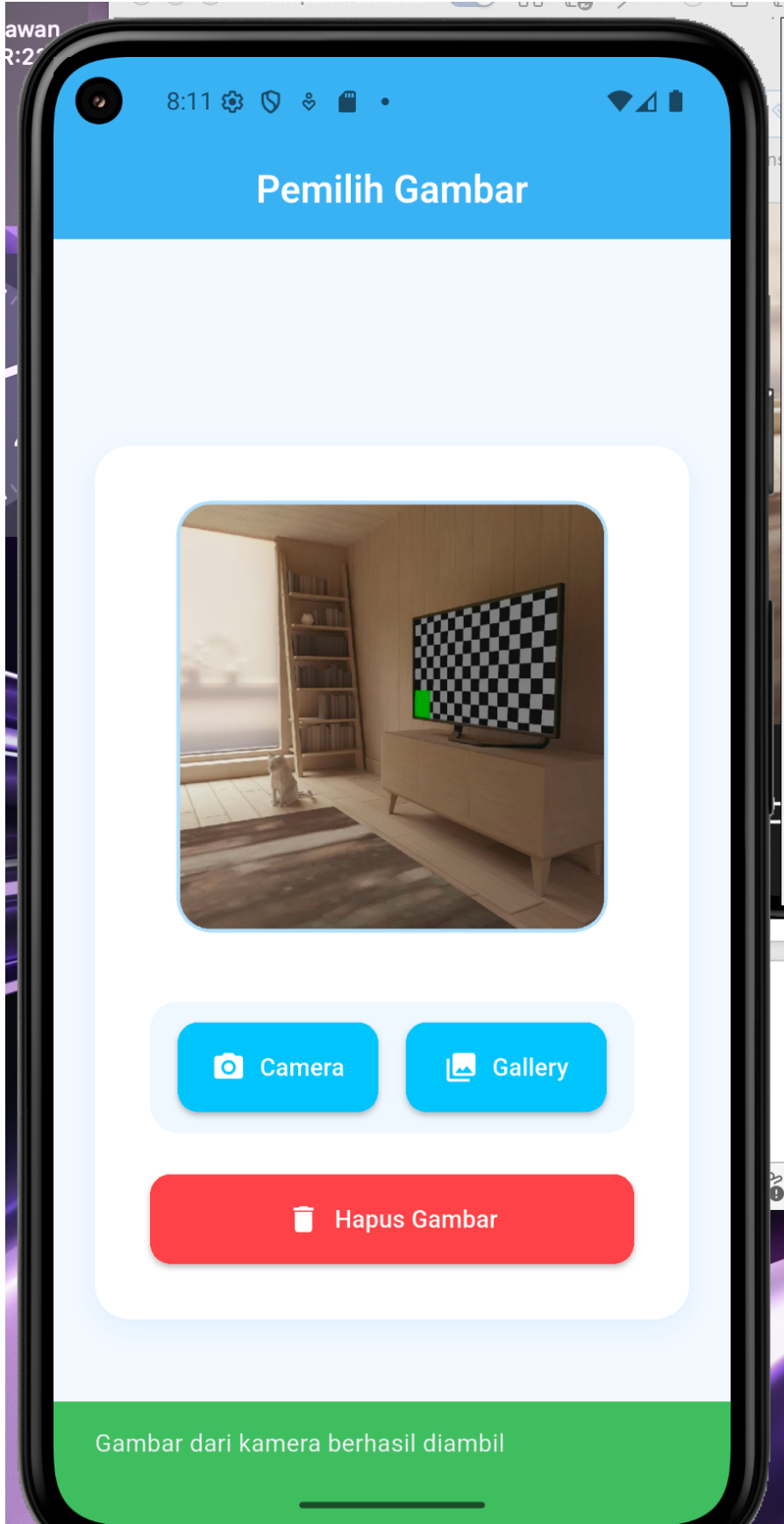
```

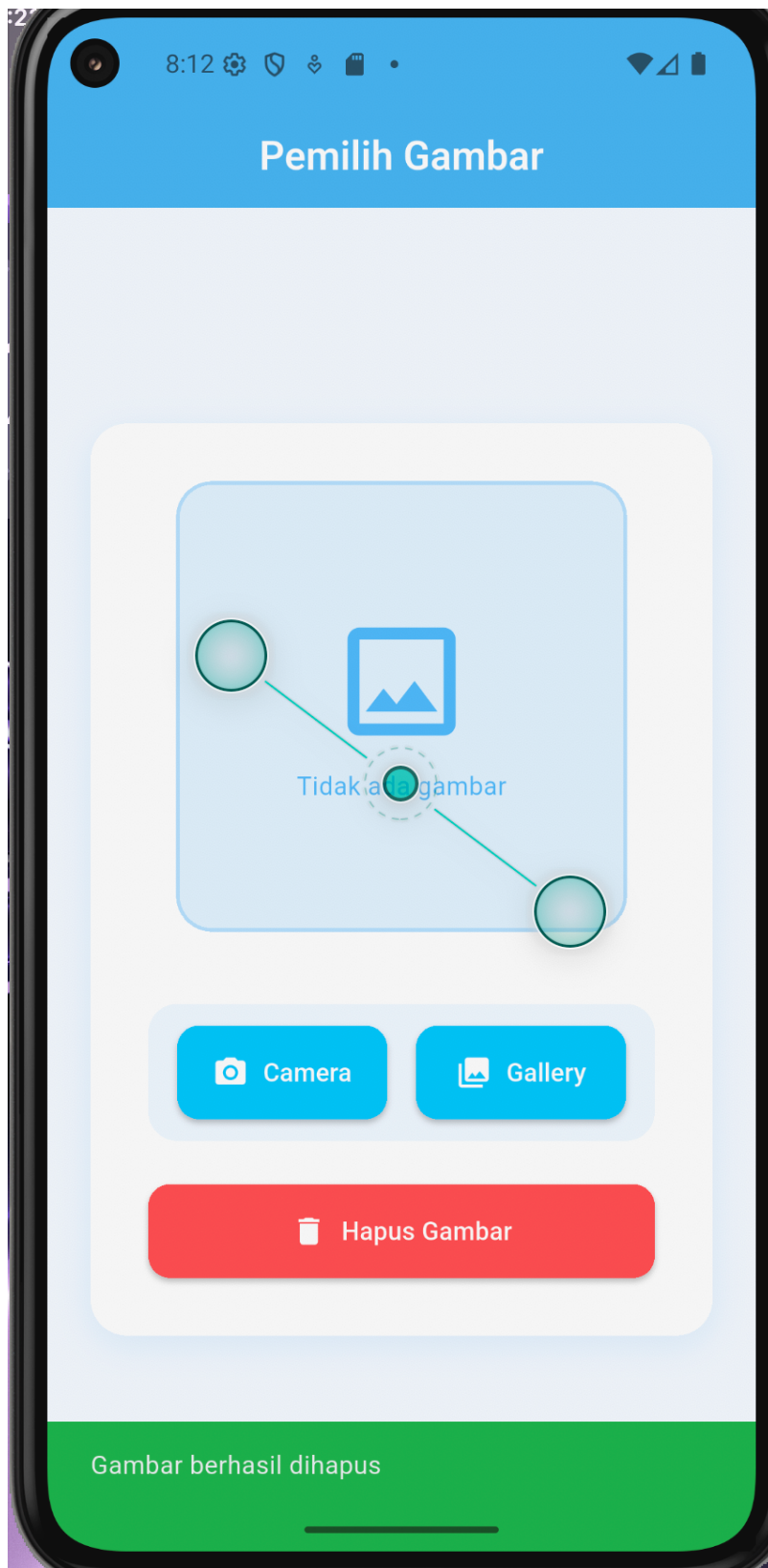
```
foregroundColor: Colors.white,  
padding: const EdgeInsets.symmetric(vertical: 16, horizontal: 8),  
shape: RoundedRectangleBorder(  
  borderRadius: BorderRadius.circular(12),  
),  
elevation: 2,  
),  
);  
}  
}
```

Output









Deskripsi Program

Secara fungsional, aplikasi ini menawarkan tiga fitur utama yang terintegrasi dengan baik. Pertama, tombol **Camera** memungkinkan pengguna mengambil foto langsung menggunakan kamera perangkat. Ketika tombol ini ditekan, aplikasi akan secara otomatis meminta izin akses kamera kepada pengguna, kemudian membuka antarmuka kamera native untuk pengambilan gambar. Setelah foto berhasil diambil, gambar tersebut langsung ditampilkan dalam container yang telah disediakan. Kedua, tombol **Gallery** memberikan akses ke galeri foto perangkat, memungkinkan pengguna memilih gambar yang sudah tersimpan. Mirip dengan fungsi kamera, aplikasi akan meminta izin akses penyimpanan terlebih dahulu sebelum menampilkan pilihan gambar dari galeri. Ketiga, tombol **Hapus Gambar** berfungsi untuk menghapus gambar yang sedang ditampilkan, dengan fitur keamanan tambahan berupa dialog konfirmasi untuk mencegah penghapusan tidak sengaja.

Dari sisi teknis, aplikasi ini mengimplementasikan beberapa package Flutter penting seperti **image_picker** untuk mengakses kamera dan galeri, serta **permission_handler** untuk mengelola permintaan izin akses perangkat. Aplikasi juga dilengkapi dengan mekanisme error handling yang komprehensif untuk menangani berbagai skenario seperti penolakan izin, kegagalan pengambilan gambar, atau masalah lainnya. Fitur loading indicator memberikan feedback visual kepada pengguna selama proses pengambilan gambar berlangsung, sementara snackbar notifications menginformasikan hasil dari setiap aksi yang dilakukan.

Arsitektur aplikasi dibangun menggunakan pendekatan state management dengan **StatefulWidget**, memastikan bahwa perubahan state (seperti gambar yang dipilih) langsung direfleksikan dalam tampilan UI. Desain antarmuka mengadopsi prinsip Material Design 3 dengan elemen-elemen visual seperti rounded corners, shadows, dan spacing yang konsisten, menciptakan tampilan yang modern dan kohesif. Aplikasi ini juga telah dikonfigurasi dengan permissions yang diperlukan untuk kedua platform utama (Android dan iOS), memastikan kompatibilitas dan fungsionalitas yang optimal di berbagai perangkat mobile.