LAPORAN PRAKTIKUM

MOBILE & WEB SERVICE PRAKTIK

Pertemuan ke-3

Semester Ganjil TA. 2024/2025

Flutter Lanjutan, Widget ListView, Widget GridView, dan Flutter Navigation

Dosen Pengampu: Suyud Widiono, S.Pd., M.Kom.

Disusun oleh :

Regita Cahya Arrahma (5220411359)

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2024

DAFTAR ISI

BAB I
FLUTTER COMPONENT LANJUTAN

1.1. Dasar Teori

1.1.1. Stack

Stack adalah widget dalam Flutter yang memungkinkan penumpukan beberapa widget child satu sama lain dalam lapisan-lapisan, mirip dengan tumpukan kartu. Setiap widget anak yang ditambahkan ke dalam Stack akan ditempatkan di atas yang sebelumnya. Salah satu keunggulan Stack adalah kemampuannya untuk memposisikan widget anak secara bebas dengan menggunakan widget Positioned, yang memungkinkan penempatan spesifik pada setiap sisi, seperti menentukan jarak dari tepi kiri, atas, kanan, atau bawah dari Stack.

Properti utama Stack adalah children, yang berisi daftar widget yang ingin Anda tumpuk. Semua widget yang dimasukkan ke dalam daftar ini akan dirender berdasarkan urutan penempatannya, di mana widget terakhir akan berada di lapisan teratas. Selain itu, Stack juga memiliki properti alignment, yang digunakan untuk menentukan bagaimana setiap widget anak harus disejajarkan dalam Stack jika tidak diposisikan secara eksplisit dengan Positioned. Alignment ini dapat berupa nilai-nilai seperti Alignment.topLeft, Alignment.center, atau lainnya yang mengontrol posisi default dari widget anak dalam Stack.

1.1.2. Padding

Padding adalah widget di Flutter yang berfungsi untuk menambahkan ruang di sekitar widget child-nya. Widget ini sangat berguna dalam pengaturan tata letak karena memungkinkan pengembang memberikan jarak antar elemen dalam sebuah layout, sehingga tampilan antarmuka menjadi lebih rapi dan terstruktur. Dengan menambahkan ruang di sekitar widget, Padding membantu mencegah elemen-elemen saling menempel atau tumpang tindih, serta meningkatkan keterbacaan dan kejelasan visual.

Properti utama dari Padding adalah padding, yang menentukan seberapa besar ruang yang ingin ditambahkan di sekitar widget child. Ruang ini dapat diatur menggunakan kelas EdgeInsets, yang menyediakan beberapa cara berbeda untuk mengatur nilai padding sesuai kebutuhan. Misalnya, EdgeInsets.all(double value) digunakan untuk memberikan padding yang sama di semua sisi widget. Jika Anda ingin memberikan padding yang berbeda untuk sisi horizontal dan vertikal, Anda

dapat menggunakan EdgeInsets.symmetric({double horizontal, double vertical}). Sedangkan untuk kasus di mana Anda hanya ingin menambahkan padding pada sisi-sisi tertentu, seperti hanya di bagian atas atau kiri, EdgeInsets.only({double left, double top, double right, double bottom}) dapat digunakan untuk memberikan kontrol lebih detail.

1.1.3. Align

Align adalah widget di Flutter yang berfungsi untuk mengatur posisi sebuah widget child di dalam ruang yang disediakan oleh Align itu sendiri. Widget ini sangat bermanfaat dalam pengaturan tata letak yang membutuhkan kontrol lebih presisi pada penempatan elemen di dalam layout yang fleksibel. Dengan Align, posisi widget anak dapat diatur di berbagai titik dalam area yang tersedia, seperti di tengah, di sudut, atau di bagian lain dari kontainer.

Properti utama dari widget ini adalah alignment, yang menentukan bagaimana dan di mana widget child ditempatkan di dalam Align. Properti alignment dapat menerima berbagai nilai dari kelas Alignment, seperti Alignment.topLeft, Alignment.center, Alignment.bottomRight, dan sebagainya. Misalnya, penggunaan Alignment.center akan menempatkan widget anak tepat di tengah area yang tersedia. Sementara itu, Alignment.bottomRight akan menempatkan widget di sudut kanan bawah.

1.1.4. Elevated Button

ElevatedButton adalah widget di Flutter yang digunakan untuk menampilkan tombol (button) yang menonjol dengan efek bayangan dan ketinggian. Widget ini merupakan salah satu komponen antarmuka pengguna yang sering digunakan untuk menangani aksi pengguna, seperti ketika tombol ditekan untuk mengeksekusi suatu fungsi atau mengirimkan data. ElevatedButton memberikan tampilan yang lebih interaktif karena efek elevasi (bayangan) yang menambah dimensi pada tombol, membuatnya tampak seperti terangkat dari permukaan.

ElevatedButton memiliki beberapa properti penting, seperti onPressed, yang berfungsi untuk menangani aksi yang terjadi ketika tombol ditekan. Jika onPressed diberikan nilai fungsi, tombol akan aktif, tetapi jika null, tombol akan dalam keadaan non-aktif (disable). Properti lainnya seperti style memungkinkan untuk mengatur tampilan tombol, termasuk warna, bentuk, dan efek bayangan. ElevatedButton juga

dapat menampung widget child, biasanya berupa teks atau ikon, yang menjadi konten visual di dalam tombol tersebut.

1.1.5. TextField

TextField adalah widget di Flutter yang digunakan untuk membuat komponen input teks, di mana pengguna dapat mengetikkan teks atau data lainnya. Widget ini sangat penting dalam pembuatan antarmuka pengguna yang interaktif, terutama untuk menerima masukan pengguna, seperti mengisi formulir, memasukkan kata sandi, atau melakukan pencarian. TextField mendukung berbagai fitur yang memudahkan pengguna, seperti penanganan input teks, validasi data, serta berbagai opsi kustomisasi tampilan dan perilaku.

Beberapa properti penting dari TextField termasuk controller, yang digunakan untuk mengelola dan mengontrol teks yang dimasukkan, serta onChanged dan onSubmitted, yang memungkinkan respons terhadap perubahan teks secara real-time atau ketika input diselesaikan oleh pengguna. Selain itu, TextField mendukung kustomisasi tampilan, seperti mengatur label teks, hint (petunjuk teks), ikon, dan gaya teks untuk menciptakan pengalaman pengguna yang sesuai dengan kebutuhan aplikasi.

1.1.6. Image

Image adalah widget di Flutter yang berfungsi untuk menampilkan gambar dalam aplikasi, baik dari sumber internet maupun dari assets lokal.

a. Internet

Ketika menggunakan Image untuk menampilkan gambar dari internet, gambar dapat dimuat menggunakan URL sebagai sumbernya. Hal ini dilakukan dengan menggunakan Image.network(), di mana URL gambar diberikan sebagai parameter. Widget ini secara otomatis akan memuat gambar dari internet dan menampilkannya di antarmuka aplikasi. Keuntungan dari menggunakan gambar dari internet adalah kemudahan dalam memperbarui konten tanpa harus memperbarui aplikasi. Namun, penting untuk memastikan bahwa koneksi internet tersedia dan gambar dapat diakses.

b. Assets lokal

Sebaliknya, untuk menampilkan gambar dari folder lokal, dapat digunakan Image.asset(). Gambar yang ingin ditampilkan harus disimpan dalam folder assets di proyek Flutter dan terdaftar dalam file pubspec.yaml. Metode ini

berguna untuk gambar yang merupakan bagian dari aplikasi, seperti ikon, logo, atau gambar latar belakang. Penggunaan gambar lokal memastikan bahwa konten selalu tersedia tanpa tergantung pada koneksi internet, serta memberikan kinerja yang lebih baik dalam beberapa kasus.

1.1.7. Container

Container adalah widget serbaguna di Flutter yang digunakan untuk mengelola tata letak, ukuran, padding, margin, border, dan background dari widget lain. Sebagai elemen dasar dalam Flutter, Container sering digunakan untuk membangun layout yang kompleks dengan menggabungkan berbagai properti untuk mengatur tampilan dan posisi widget lain di dalamnya. Fungsinya sangat fleksibel, sehingga Container dapat menampung dan mengatur satu widget child, memberikan berbagai pengaturan visual, sekaligus menyesuaikan tata letak di sekitarnya.

Properti utama dari Container meliputi:

- Width & Height: Properti ini digunakan untuk menentukan lebar dan tinggi dari container. Jika tidak diatur, ukuran container akan menyesuaikan dengan ukuran child di dalamnya atau sesuai dengan batas-batas lain yang diterapkan oleh parent widget.
- Padding: Padding memberikan ruang di dalam container antara konten dan tepi container. Padding membantu dalam menambahkan jarak internal untuk konten di dalam container, membuat tata letak lebih teratur.
- Margin: Margin menambahkan ruang di luar container antara container dan widget lain. Ini membantu dalam memberikan jarak eksternal antara Container dan elemen-elemen di sekitarnya, sehingga tata letak menjadi lebih rapi.
- Decoration: Properti ini memungkinkan untuk menambahkan dekorasi seperti warna latar belakang, border, radius sudut (corner radius), bayangan (shadow), dan gambar latar belakang. Decoration dapat digunakan untuk mempercantik tampilan Container dan memberikan efek visual yang menarik.

1.1.8. Icon

Icon adalah widget di Flutter yang berfungsi untuk menampilkan ikon grafis. Ikon ini biasanya digunakan sebagai elemen visual yang mewakili tindakan, status, atau informasi tertentu dalam sebuah antarmuka pengguna. Icon sering digunakan dalam tombol, menu, atau bagian lain dari aplikasi untuk memperjelas fungsi atau tindakan yang dapat dilakukan pengguna, seperti ikon pencarian, pengaturan, atau favorit.

1.2. Praktikum

1.2.1. Stack

Berikut ini adalah kode program penggunaan Stack:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const StackPositionPage(),
    );
  }
}

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

  @override
  State<StackPositionPage> createState() => _StackPositionedPageState();
}

class _StackPositionedPageState extends State<StackPositionPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Stack & Positioned',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: Stack(
        children: [
          Center(
            child: Container(
```

```
            width: 100.0,
            height: 100.0,
            color: Colors.blue,
          ),
        ),
        Positioned(
          left: 0.0,
          right: 0.0,
          top: 0.0,
          bottom: 0.0,
          child: Center(
            child: Container(
              width: 50.0,
              height: 50.0,
              color: Colors.pink,
            ),
          ),
        ),
        Positioned(
          left: 0.0,
          top: 0.0,
          child: Center(
            child: Container(
              width: 50.0,
              height: 50.0,
              color: Colors.pink,
            ),
          ),
        ),
        Positioned(
          right: 0.0,
          bottom: 0.0,
          child: Center(
            child: Container(
              width: 100.0,
              height: 100.0,
              color: Colors.orange,
            ),
          ),
        ),
        Positioned(
          right: 0.0,
          bottom: 0.0,
          child: Center(
            child: Container(
              width: 50.0,
              height: 50.0,
              color: Colors.blue,
```

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

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `StackPositionPage`.

4. `class StackPositionPage extends StatefulWidget {…}`

   `StackPositionPage` adalah widget `StatefulWidget`, yang berarti widget ini dapat berubah seiring waktu. Kelas ini mendefinisikan tampilan layar yang menggunakan widget `Stack`.

5. `State<StackPositionPage> createState()=>_StackPositionedPageState();`

Metode ini membuat dan mengembalikan `state_StackPositionedPageState`, yang bertanggung jawab untuk membangun UI dan merespon perubahan.

6. `class _StackPositionedPageState extends State<StackPositionPage> {`

   Ini adalah state dari `StackPositionPage`, di mana logika dan tampilan layar didefinisikan.

   `Widget build(BuildContext context) {`

   Metode ini membangun UI menggunakan beberapa widget utama:

   `Scaffold`: Struktur dasar layar Flutter yang menyediakan API standar untuk UI Material. Di sini, `Scaffold` memiliki:

   - `appBar`: AppBar dengan judul 'Stack & Positioned'.
   - `body`: Isi utama layar diatur dalam widget `SafeArea` untuk menghindari area notifikasi dan batas layar.

7. Stack dan Positioned (layout di dalam body)

   `Stack` adalah widget yang memungkinkan anak-anaknya saling ditumpuk di atas satu sama lain. Setiap elemen di dalam `Stack` dapat ditempatkan dengan presisi menggunakan widget `Positioned`.

   Elemen-elemen di dalam Stack:

   1. `Center` dengan `Container` biru (Blue box)**:**
      - Ditempatkan di tengah layar menggunakan `Center` widget.
      - `Container`: Menghasilkan kotak dengan ukuran 100x100 pixel dan warna biru.
   2. `Positioned` (Pink box di tengah)**:**
      - `Positioned` digunakan untuk menempatkan widget di lokasi tertentu dalam `Stack`.
      - Di sini, `Positioned` menempatkan sebuah `Container` berukuran 50x50 pixel di tengah layar (secara eksplisit mengisi seluruh sisi dengan nilai `left`, `right`, `top`, dan `bottom` bernilai `0.0`).
   3. `Positioned` (Pink box di kiri atas)**:**

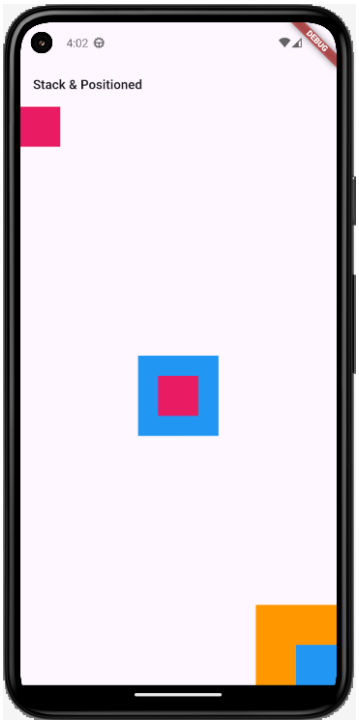
      Kotak lain berwarna pink ditempatkan di kiri atas layar (`left: 0.0`, `top: 0.0`), menggunakan `Center` yang sebenarnya tidak diperlukan karena widget ditempatkan pada posisi tertentu.
   4. `Positioned` (Orange box di kanan bawah)**:**

Kotak berukuran 100x100 pixel dengan warna oranye ditempatkan di kanan bawah layar menggunakan `Positioned(right: 0.0, bottom: 0.0)`.

5. `Positioned` (Blue box kecil di kanan bawah)**:**

   Di sini, kotak berwarna biru berukuran 50x50 ditempatkan di atas kotak oranye, di posisi yang sama (kanan bawah), sehingga akan terlihat tumpang tindih dengan kotak oranye.

Hasil running dari kode program:



1.2.2. Padding

Berikut ini adalah kode program penggunaan Padding:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
```

```
      ),
      home: const PaddingPage(),
    );
  }
}

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

  @override
  State<PaddingPage> createState() => _PaddingPageState();
}

class _PaddingPageState extends State<PaddingPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Padding',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
            child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Material(
              color: Colors.red.shade100,
              child: const Text('Widget Tanpa Padding'),
            ),
            const SizedBox(
              height: 16.0,
            ),
            Material(
              color: Colors.yellow.shade100,
              child: const Padding(
                padding: EdgeInsets.all(16.0),
                child: Text(
                  'Widget Dengan Padding',
                ),
              ),
            ),
            const SizedBox(
              height: 16.0,
            ),
            Material(
```

```
              color: Colors.green.shade100,
              child: const Padding(
                padding: EdgeInsets.symmetric(horizontal: 32.0, vertical:
8.0),
                child: Text('Widget Dengan Padding'),
              ),
            ),
            const SizedBox(
              height: 16.0,
            ),
            Material(
              color: Colors.blue.shade100,
              child: const Padding(
                padding: EdgeInsets.only(
                    left: 32.0, right: 16.0, top: 16.0, bottom: 8.0),
                child: Text('Widget Dengan Padding'),
              ),
            ),
          ],
        )),
      ),
    );
  }
}
```

Penjelasan dari setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.

- `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
- `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `PaddingPage`.

4. `class PaddingPage extends StatefulWidget {…}`

   Kelas `PaddingPage` adalah widget stateful yang mendefinisikan tampilan halaman dengan contoh penggunaan padding.

5. `class _PaddingPageState extends State<PaddingPage> {…}`

   Kelas `_PaddingPageState` adalah state dari widget `PaddingPage`, yang berisi UI untuk halaman tersebut.

   `Widget build(BuildContext context) {…}`

   Metode ini membangun antarmuka pengguna untuk halaman `PaddingPage`. Komponen utama adalah:

   `Scaffold`: Struktur dasar untuk halaman, menyediakan API standar untuk elemen-elemen UI Material. Ini mencakup:

   - `appBar`: Bar aplikasi di bagian atas dengan judul "`Padding`".
   - `body`: Isi halaman, yang menggunakan `SafeArea` untuk menghindari area di luar batas layar dan Center untuk memusatkan isinya.

6. Penggunaan Padding di dalam `body`:

   Komponen utama di dalam `body` adalah kolom yang terdiri dari beberapa widget `Material`, masing-masing menunjukkan cara kerja `Padding`. Komponen-komponen di dalam `Column`:

   1. Widget tanpa padding**:**

      `Material(color: Colors.red.shade100)`: Sebuah widget `Material` dengan latar belakang merah muda dan teks "Widget Tanpa Padding". Ini tidak memiliki `Padding`.

   2. Widget dengan padding seragam (semua sisi)**:**

      `Material(color: Colors.yellow.shade100)`: Widget ini menggunakan `Padding` dengan nilai `EdgeInsets.all(16.0)`, yang berarti padding 16 piksel diterapkan di semua sisi.

   3. Widget dengan padding simetris**:**

`Material(color: Colors.green.shade100)`: Padding di sini menggunakan `EdgeInsets.symmetric(horizontal: 32.0, vertical: 8.0)`, yang berarti padding horizontal sebesar 32 piksel dan vertikal 8 piksel.

4. Widget dengan padding khusus (per sisi)**:**

   `Material(color: Colors.blue.shade100)`: Padding diatur secara spesifik untuk setiap sisi menggunakan `EdgeInsets.only(left: 32.0, right: 16.0, top: 16.0, bottom: 8.0)`.

Hasil running dari kode program:

1.2.3. Align

Berikut ini adalah kode program penggunaan Align:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
```

```
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const AlignPage(),
    );
  }
}

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

  @override
  State<AlignPage> createState() => _AlignPageState();
}

class _AlignPageState extends State<AlignPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Align',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: Stack(
        children: [
          Padding(
            padding: const EdgeInsets.all(16.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Align(
                  alignment: Alignment.center,
                  child: Container(
                    width: 50.0,
                    height: 50.0,
                    color: Colors.green,
                  ),
                ),
                const SizedBox(
                  height: 16.0,
                ),
                Align(
                  alignment: Alignment.centerLeft,
```

```
                child: Container(
                  width: 50.0,
                  height: 50.0,
                  color: Colors.yellow,
                ),
              ),
              const SizedBox(
                height: 16.0,
              ),
              Align(
                alignment: Alignment.centerRight,
                child: Container(
                  width: 50.0,
                  height: 50.0,
                  color: Colors.blue,
                ),
              ),
            ],
          ),
        ),
        Align(
          alignment: Alignment.topLeft,
          child: Container(
            width: 50.0,
            height: 50.0,
            color: Colors.red,
          ),
        ),
        Align(
          alignment: Alignment.bottomRight,
          child: Container(
            width: 50.0,
            height: 50.0,
            color: Colors.purple,
          ),
        ),
      ],
    )),
  );
 }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `AlignPage`.

4. `class AlignPage extends StatefulWidget {…}`

   `AlignPage` adalah widget stateful yang akan digunakan untuk menampilkan contoh penggunaan widget `Align`.

5. `class _AlignPageState extends State<AlignPage> {…}`

   Kelas `_AlignPageState` adalah state dari `AlignPage`, tempat logika dan UI ditentukan.

   `Widget build(BuildContext context) {`

   Metode ini membangun UI halaman dengan menggunakan `Scaffold`, yang menyediakan struktur dasar tampilan layar.

   `Scaffold`: Wadah layar yang menyediakan elemen-elemen dasar, seperti `AppBar` dan `body`.

   - `appBar`: Bar aplikasi dengan judul "`Align`".
   - `body`: Isi halaman yang menggunakan widget Stack untuk menumpuk widget-widget di atas satu sama lain.

6. Penggunaan `Align` di dalam `body` (Tumpukan Widget):

   Dalam `Stack`, beberapa widget ditempatkan menggunakan widget `Align` untuk menyesuaikan posisi mereka. Berikut elemen-elemen di dalamnya:

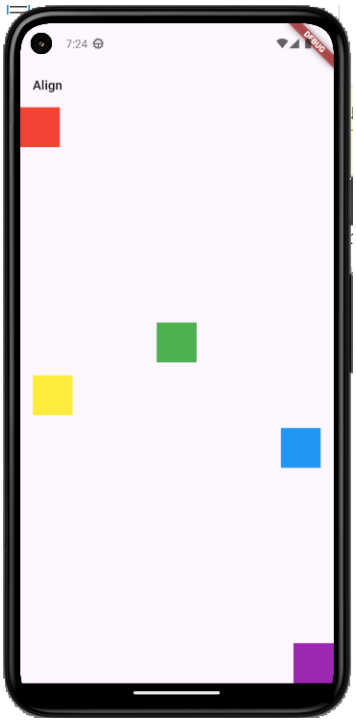
Di dalam `Column`:

1. Align(center):
    - `Align(alignment: Alignment.center)`: Posisi widget berada di tengah.
    - `Container`: Menampilkan kotak hijau berukuran 50x50 piksel.
2. Align(centerLeft):
    - `Align(alignment: Alignment.centerLeft)`: Widget diposisikan di tengah pada sumbu vertikal, tetapi di kiri pada sumbu horizontal.
    - `Container`: Menampilkan kotak kuning berukuran 50x50 piksel.
3. Align(centerRight):
    - `Align(alignment: Alignment.centerRight)`: Widget diposisikan di tengah pada sumbu vertikal, tetapi di kanan pada sumbu horizontal.
    - `Container`: Menampilkan kotak biru berukuran 50x50 piksel.

Di luar `Column`:

1. Align(topLeft):
    - `Align(alignment: Alignment.topLeft)`: Widget diposisikan di pojok kiri atas layar.
    - `Container`: Menampilkan kotak merah berukuran 50x50 piksel.
2. Align(bottomRight):
    - `Align(alignment: Alignment.bottomRight)`: Widget diposisikan di pojok kanan bawah layar.
    - `Container`: Menampilkan kotak ungu berukuran 50x50 piksel.

Hasil running dari kode program:



1.2.4. Elevated Button

Berikut ini adalah kode program penggunaan Elevated Button:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ElevatedButtonPage(),
    );
  }
}

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

```dart
  @override
  State<ElevatedButtonPage> createState() => _ElevatedButtonPageState();
}

class _ElevatedButtonPageState extends State<ElevatedButtonPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Elevated Button',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
            child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              ElevatedButton(
                  onPressed: () => log('Button clicked'),
                  style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.red,
                      foregroundColor: Colors.white),
                  child: const Text('Click Me')),
              const SizedBox(height: 16.0),
              ElevatedButton.icon(
                  onPressed: () => log('Button clicked'),
                  style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.green,
                      foregroundColor: Colors.white),
                  icon: const Icon(Icons.play_arrow_rounded),
                  label: const Text('Click Me')),
              const SizedBox(
                height: 16.0,
              ),
              SizedBox(
                width: double.maxFinite,
                height: 48.0,
                child: ElevatedButton.icon(
                  onPressed: () => log('Button clicked'),
                  style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.blue,
                      foregroundColor: Colors.white,
                      shape: RoundedRectangleBorder(
```

```
                        borderRadius: BorderRadius.circular(8.0))),
                icon: const Icon(Icons.play_arrow_rounded),
                label: const Text('Click Me'),
              ),
            ),
          ],
        ),
      )),
    ),
  );
 }
}
```

Penjelasan setiap blok dari kode program di atas:

1. `import 'dart:developer';`
   Mengimpor pustaka `dart:developer`, yang memungkinkan untuk menampilkan log atau informasi debugging di konsol menggunakan fungsi `log()`.
2. `import 'package:flutter/material.dart';`
   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.
3. `void main() { runApp(const MyApp());}`
   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).
4. `class MyApp extends StatelessWidget {…}`
   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).
   `Widget build(BuildContext context) {`
   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.
   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.

- `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ElevatedButtonPage`.

5. `class ElevatedButtonPage extends StatefulWidget {…}`
   `ElevatedButtonPage` adalah widget stateful yang akan menampilkan contoh-contoh penggunaan `ElevatedButton`.
6. `class _ElevatedButtonPageState extends State<ElevatedButtonPage> {`
   Kelas `_ElevatedButtonPageState` bertanggung jawab untuk membangun UI halaman `ElevatedButtonPage`.

   `Widget build(BuildContext context) {`
   Metode ini membangun antarmuka halaman yang terdiri dari beberapa variasi `ElevatedButton`.

   `Scaffold`: Struktur dasar layar, yang menyediakan API standar untuk elemen UI Material seperti `AppBar` dan `body`.

   - `appBar`: Bar aplikasi di bagian atas dengan judul "`Elevated Button`".
   - `body`: Isi utama halaman yang menggunakan `SafeArea` untuk memastikan isi tidak terpotong oleh status bar atau bagian layar lain.

7. Penggunaan `ElevatedButton` di dalam `body`**:**

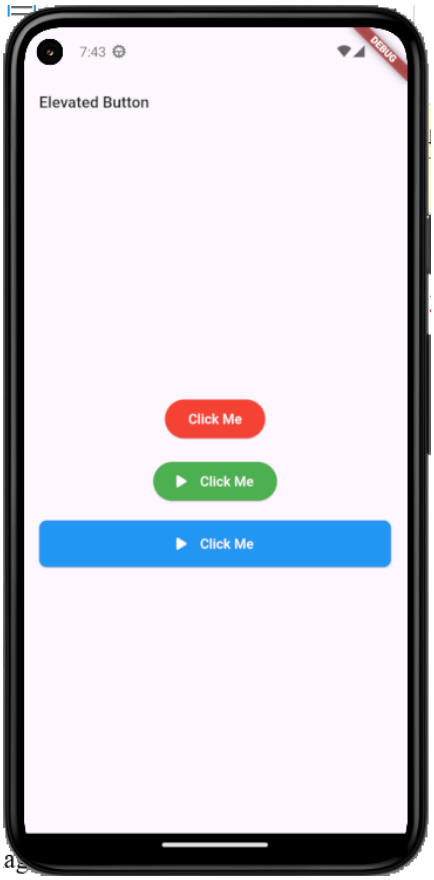
   Terdapat beberapa variasi tombol `ElevatedButton` yang digunakan di dalam `Column`, dengan setiap tombol memiliki gaya dan konfigurasi berbeda:

   1. Tombol Elevated Button sederhana:
      - `ElevatedButton(onPressed: () => log('Button clicked'))`: Tombol ini menggunakan callback `onPressed` untuk menampilkan log 'Button clicked' saat ditekan.
      - `style: ElevatedButton.styleFrom(...)`: Gaya tombol disesuaikan dengan latar belakang merah (`backgroundColor: Colors.red`) dan teks berwarna putih (`foregroundColor: Colors.white`).
      - `child: Text('Click Me')`: Konten dari tombol adalah teks "Click Me".
   2. Elevated Button dengan ikon:
      - `ElevatedButton.icon()`: Tombol ini memiliki ikon di samping teks.
      - `icon: Icon(Icons.play_arrow_rounded)`: Menggunakan ikon panah `play_arrow_rounded` dari Flutter Material.

- `label: Text('Click Me')`: Teks yang ditampilkan adalah "Click Me".
- `style: ElevatedButton.styleFrom(backgroundColor: Colors.green)`: Tombol ini memiliki warna latar belakang hijau.

3. Elevated Button dengan lebar penuh:
    - `SizedBox(width: double.maxFinite, height: 48.0)`: Menggunakan `SizedBox` untuk menetapkan lebar tombol agar memenuhi lebar layar dan tinggi tombol menjadi 48 piksel.
    - `ElevatedButton.icon()`: Tombol ini juga memiliki ikon dan label, serta gaya yang mirip dengan tombol sebelumnya.
    - `shape: RoundedRectangleBorder(...)`: Gaya tombol menggunakan `RoundedRectangleBorder` dengan sudut membulat (radius 8 piksel).

Hasil running dari kode program:



1.2.5. TextField

Berikut ini adalah kode program penggunaan TextField:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const TextFieldPage(),
    );
  }
}

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

  @override
  State<TextFieldPage> createState() => _TextFieldPageState();
}

class _TextFieldPageState extends State<TextFieldPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'TextField',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
            child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 16.0),
          child: Column(
```

```
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const TextField(
              decoration: InputDecoration(
                labelText: 'Full Name',
                hintText: 'Enter full name',
              ),
              maxLength: 50,
            ),
            const SizedBox(
              height: 16.0,
            ),
            const TextField(
              decoration: InputDecoration(
                  labelText: 'Phone Number',
                  hintText: 'Enter phone number',
                  filled: true),
              keyboardType: TextInputType.number,
              maxLength: 13,
            ),
            const TextField(
              decoration: InputDecoration(
                labelText: 'Email',
                hintText: 'Enter email address',
                border: OutlineInputBorder(),
                prefixIcon: Icon(Icons.email_rounded),
              ),
              keyboardType: TextInputType.emailAddress,
              maxLength: 50,
            ),
            const SizedBox(
              height: 16.0,
            ),
            TextField(
              obscureText: true,
              decoration: InputDecoration(
                  labelText: 'Password',
                  hintText: 'Enter Password',
                  border: const OutlineInputBorder(),
                  prefixIcon: const Icon(Icons.password_rounded),
                  suffixIcon: IconButton(
                    icon: const Icon(Icons.visibility_off_rounded),
                    onPressed: () => log('Update password visibility'),
                  )),
              maxLength: 20,
            ),
          ],
        ),
```

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

Penjelasan setiap blok dari kode program di atas:

1. `import 'dart:developer';`

   Mengimpor pustaka `dart:developer`, yang memungkinkan untuk menampilkan log atau informasi debugging di konsol menggunakan fungsi `log()`.

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

3. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

4. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `TextFieldPage`.

5. `class TextFieldPage extends StatefulWidget {…}`

   `TextFieldPage` adalah widget stateful yang akan digunakan untuk menampilkan beberapa input menggunakan `TextField`.

6. `class _TextFieldPageState extends State<TextFieldPage> {`

Kelas `_TextFieldPageState` bertanggung jawab untuk membangun UI halaman `TextFieldPage`.

```
Widget build(BuildContext context) {..}
```

Metode ini membangun antarmuka halaman yang terdiri dari beberapa `TextField` dengan berbagai properti.

`Scaffold`: Struktur dasar layar, yang menyediakan API standar untuk elemen UI Material seperti `AppBar` dan `body`.

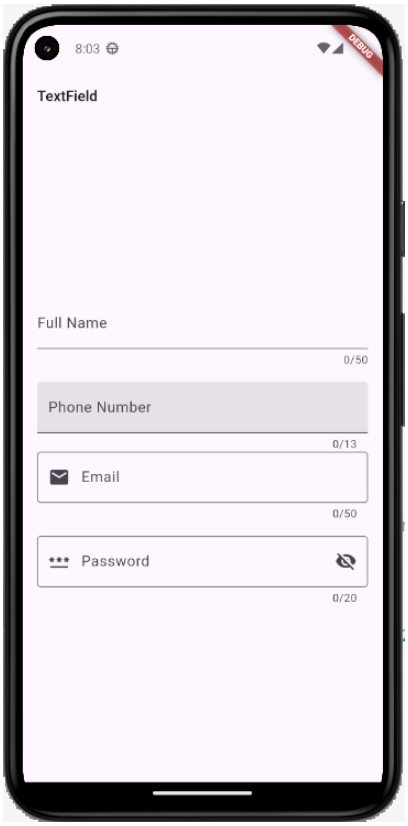
- `appBar`: Bar aplikasi di bagian atas dengan judul "TextField".
- `body`: Isi utama halaman, yang menggunakan `SafeArea` untuk memastikan konten tidak terpotong oleh bagian layar lain.

7. Penggunaan `TextField` di dalam `body`:
   Setiap `TextField` memiliki dekorasi dan properti yang berbeda untuk menyesuaikan dengan jenis input yang diharapkan.
   1. TextField untuk Nama Lengkap**:**
      - `TextField(decoration: ...)`: Membuat input teks untuk nama lengkap.
      - `labelText: 'Full Name'`: Menampilkan label "Full Name" di atas input.
      - `hintText: 'Enter full name'`: Menampilkan placeholder di dalam input sebagai petunjuk.
      - `maxLength: 50`: Membatasi jumlah karakter maksimal menjadi 50.
   2. TextField untuk Nomor Telepon**:**
      - `keyboardType: TextInputType.number`: Menampilkan keyboard angka untuk memudahkan memasukkan nomor telepon.
      - `maxLength: 13`: Jumlah karakter maksimal untuk nomor telepon adalah 13.
      - `filled: true`: Memberikan warna latar belakang pada input.
   3. TextField untuk Email**:**
      - `TextField(decoration: ...)`: Membuat input teks untuk alamat email.
      - `labelText: 'Email'`: Menampilkan label "Email" di atas input.
      - `hintText: 'Enter email address'`: Menampilkan placeholder di dalam input.

- `border: OutlineInputBorder()`: Menambahkan batas (outline) di sekitar input.
- `prefixIcon: Icon(Icons.email_rounded)`: Menambahkan ikon email di dalam input.
- `keyboardType: TextInputType.emailAddress`: Menampilkan keyboard khusus untuk email.
- `maxLength: 50`: Membatasi input maksimal menjadi 50 karakter.

4. TextField untuk Password:
    - `obscureText: true`: Mengaktifkan mode tersembunyi untuk input password (karakter tidak terlihat).
    - `labelText: 'Password'`: Menampilkan label "Password" di atas input.
    - `hintText: 'Enter Password'`: Menampilkan placeholder di dalam input.
    - `border: OutlineInputBorder()`: Menambahkan batas (outline) di sekitar input.
    - `prefixIcon: Icon(Icons.password_rounded)`: Menambahkan ikon password di dalam input.
    - `suffixIcon: IconButton(...)`: Menambahkan ikon untuk mengubah visibilitas password (ikon mata). Callback untuk ikon ini menggunakan `log()` untuk mencatat perubahan visibilitas password.
    - `maxLength: 20`: Jumlah maksimal karakter password adalah 20.

Hasil running dari kode program:



1.2.6. Image

a. Internet

Berikut ini adalah kode program penggunaan Image internet:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ImageNetworkPage(),
    );
```

```
  }
}

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

  @override
  State<ImageNetworkPage> createState() => _ImageNetworkPageState();
}

class _ImageNetworkPageState extends State<ImageNetworkPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Image Network',
          style: TextStyle(fontSize: 16.0, fontWeight:
FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
            child: ClipRRect(
          borderRadius: BorderRadius.circular(8.0),
          child: Image.network(
            'https://siska.fppti.or.id/storage/foto/8LGeefnhw6FrzpKIP
ORN8RjdZgmSoLJ5djFcoUXA.png',
            width: 300.0,
            height: 300.0,
            fit: BoxFit.cover,
            filterQuality: FilterQuality.medium,
          ),
        )),
      ),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   ```
   Widget build(BuildContext context) {
   ```

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ImageNetworkPage`.

4. `class ImageNetworkPage extends StatefulWidget {…}`

   Widget ini adalah halaman yang menampilkan gambar yang diambil dari internet. StatefulWidget dipilih karena widget ini memungkinkan perubahan status jika diperlukan.

5. `class _ImageNetworkPageState extends State<ImageNetworkPage> {…}`

   Di dalam kelas ini, UI dari halaman `ImageNetworkPage` dibangun menggunakan metode `build()`.

6. `Widget build(BuildContext context) {`

   Metode ini membangun UI dari halaman.

   - `Scaffold`: Struktur dasar layar yang menyediakan elemen-elemen UI Material, seperti `AppBar` dan `body`.
     - `AppBar`: Bar aplikasi di bagian atas dengan judul "Image Network".
     - `SafeArea`: Memastikan konten halaman tidak terpotong oleh bagian-bagian layar yang mungkin mengganggu, seperti status bar atau notch.

- `Center`: Widget ini memusatkan konten yang ada di dalamnya.
- `ClipRRect`: Menyediakan kemampuan untuk memotong (clip) widget dengan sudut melengkung (dalam hal ini, gambar memiliki sudut melengkung dengan radius 8.0).
- `Image.network`: Widget ini digunakan untuk menampilkan gambar dari URL. Argumen yang digunakan:
  - `width`: Lebar gambar ditetapkan menjadi 300 piksel.
  - `height`: Tinggi gambar juga 300 piksel.
  - `fit: BoxFit.cover`: Menentukan bagaimana gambar akan diisi ke dalam widget; `BoxFit.cover` akan membuat gambar menutupi seluruh ruang yang tersedia, namun tetap mempertahankan rasio aspeknya.
  - `filterQuality: FilterQuality.medium`: Menentukan kualitas filter gambar, sehingga gambar yang diunduh dapat terlihat lebih halus.

Hasil running dari kode program:

- Assets lokal

  Berikut ini adalah kode program penggunaan Image assets lokal:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ImageAssetPage(),
    );
  }
}

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

  @override
  State<ImageAssetPage> createState() => _ImageAssetPageState();
}

class _ImageAssetPageState extends State<ImageAssetPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Image Asset',
          style: TextStyle(fontSize: 16.0, fontWeight:
FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
            child: ClipRRect(
          borderRadius: BorderRadius.circular(8.0),
```

```
          child: Image.asset(
            'assets/images/cat.jpeg',
            width: 300.0,
            height: 300.0,
            fit: BoxFit.cover,
            filterQuality: FilterQuality.medium,
          ),
        )),
      ),
    );
  }
}
```

Penjelasan setiap blok pada kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

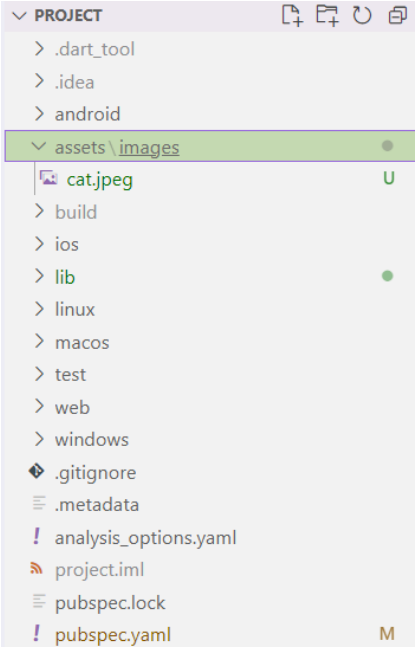
   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ImageAssetPage`.

4. `class ImageAssetPage extends StatefulWidget {…}`
   Ini adalah halaman yang dapat berubah (stateful) yang menampilkan gambar dari asset lokal.
5. `class _ImageAssetPageState extends State<ImageAssetPage> {…}`
   Kelas ini menyimpan state dari halaman `ImageAssetPage` dan bertanggung jawab untuk membangun UI di halaman tersebut.
6. `Widget build(BuildContext context) {…}`
   Metode ini membangun antarmuka pengguna untuk halaman `ImageAssetPage`.
   - `Scaffold`: Menyediakan struktur dasar untuk halaman, dengan `AppBar` dan body.
     - `AppBar`: Menampilkan bar aplikasi dengan teks "Image Asset".
     - `SafeArea`: Memastikan bahwa konten halaman tidak tertimpa bagian layar yang sensitif, seperti notch atau status bar.
     - `Center`: Menyelaraskan konten ke tengah layar.
     - `ClipRRect`: Digunakan untuk memberikan sudut yang melengkung (rounded corners) pada widget yang dibungkusnya, dalam hal ini gambar.
     - `Image.asset`: Widget ini digunakan untuk menampilkan gambar yang disimpan di direktori lokal aplikasi (dalam folder `assets`).
       - `'assets/images/cat.jpeg'`: Jalur menuju file gambar yang ditampilkan. Pastikan bahwa gambar tersebut sudah ditambahkan ke dalam file `pubspec.yaml` agar dapat diakses sebagai asset.
       - `width`: Lebar gambar ditetapkan sebesar 300 piksel.
       - `height`: Tinggi gambar juga ditetapkan sebesar 300 piksel.
       - `fit: BoxFit.cover`: Menyesuaikan ukuran gambar sehingga memenuhi seluruh ruang yang tersedia, dengan tetap menjaga rasio aspeknya.

- **`filterQuality: FilterQuality.medium`**: Menentukan kualitas filter untuk menghasilkan gambar yang lebih halus saat diskalakan.

Langkah untuk menambahkan gambar assets:

a. Buat folder baru assets/images posisi folder sejajar dengan pubspec.yaml.



b. Masukkan gambar ke dalam folder tersebut

c. Registrasikan gambar yang tadi dimasukkan dengan cara edit pubspec.yaml.

```
# To add assets to your application, add an assets section, like this
# assets:
#   - images/a_dot_burr.jpeg
#   - images/a_dot_ham.jpeg
  assets:
    - assets/images/cat.jpeg
```

Hasil running dari kode program:

1.2.7. Container

Berikut ini adalah kode program penggunaan Container:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ContainerPage(),
    );
  }
}
```

```dart
class ContainerPage extends StatefulWidget {
  const ContainerPage({super.key});

  @override
  State<ContainerPage> createState() => _ContainerPageState();
}

class _ContainerPageState extends State<ContainerPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Container',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: Center(
        child: Container(
          width: 200.0,
          height: 200.0,
          padding: const EdgeInsets.all(16.0),
          margin: const EdgeInsets.all(16.0),
          decoration: BoxDecoration(
              color: Colors.pink,
              borderRadius: BorderRadius.circular(8.0),
              boxShadow: const [
                BoxShadow(
                    color: Colors.black26,
                    blurRadius: 12.0,
                    spreadRadius: 4.0,
                    offset: Offset(0.0, 4.0))
              ]),
          child: const Center(
            child: Text(
              'Lorem ipsum dolor sit amet, consectur adipiscing elit',
              style: TextStyle(
                fontSize: 20.0,
                color: Colors.white,
                fontWeight: FontWeight.w600,
              ),
              textAlign: TextAlign.center,
            ),
          ),
        ),
      )),
```

```
    );
  }
}
```

Penjelasan setiap blok pada kode programm di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ContainerPage`.

4. `class ContainerPage extends StatefulWidget {…}`

   `ContainerPage` adalah widget yang bersifat stateful, artinya widget ini dapat berubah selama siklus hidup aplikasi.

5. `class _ContainerPageState extends State<ContainerPage> {…}`

   Ini adalah state dari `ContainerPage` yang mengatur dan membangun UI halaman tersebut.

6. `Widget build(BuildContext context) {…}`

   Metode ini membangun antarmuka pengguna untuk halaman `ContainerPage`.

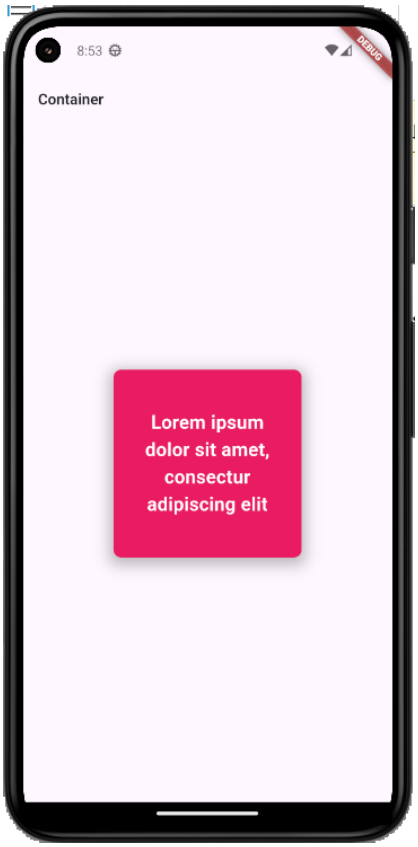
   - `Scaffold`: Menyediakan struktur halaman dasar.

- `AppBar`: Menampilkan bar aplikasi dengan teks "Container".
- `SafeArea`: Memastikan bahwa konten berada dalam area yang aman (tidak terhalang oleh status bar, notch, dll).
- `Center`: Menyelaraskan konten ke tengah layar.
- `Container`: Widget utama yang digunakan untuk menampilkan sebuah kotak yang memiliki berbagai properti seperti:
  - `width: 200.0` dan `height: 200.0`: Menentukan ukuran kontainer.
  - `padding: EdgeInsets.all(16.0)`: Memberikan ruang di dalam kontainer antara tepi kontainer dan child-nya.
  - `margin: EdgeInsets.all(16.0)`: Memberikan jarak antara kontainer dengan widget di sekitarnya.
  - `decoration: BoxDecoration`: Mendekorasi kontainer dengan:
    - `color: Colors.pink`: Memberikan warna latar belakang kontainer.
    - `borderRadius: BorderRadius.circular(8.0)`: Membuat sudut kontainer melengkung sebesar 8 piksel.
    - `boxShadow`: Memberikan bayangan di belakang kontainer dengan:
      - `color: Colors.black26`: Warna bayangan hitam dengan transparansi 26%.
      - `blurRadius: 12.0`: Seberapa lembut atau buram bayangan.
      - `spreadRadius: 4.0`: Seberapa jauh bayangan menyebar.
      - `offset: Offset(0.0, 4.0)`: Posisi bayangan, di mana sumbu X diatur ke 0.0 (tidak bergerak horizontal) dan

sumbu Y diatur ke 4.0 (bergerak vertikal ke bawah).

- `child: Center`: Widget anak yang berada di tengah kontainer.
    - `Text`: Menampilkan teks "Lorem ipsum dolor sit amet, consectur adipiscing elit" di tengah kontainer.
        - `style`: Mengatur gaya teks:
            - `fontSize: 20.0`: Ukuran font.
            - `color: Colors.white`: Warna teks putih.
            - `fontWeight: FontWeight.w600`: Ketebalan teks (medium).
        - `textAlign: TextAlign.center`: Menyelaraskan teks di tengah.

Hasil running dari kode program:



1.2.8. Icon

Berikut ini adalah kode program penggunaan Icon:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const IconPage(),
    );
  }
}

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

  @override
  State<IconPage> createState() => _IconPageState();
}

class _IconPageState extends State<IconPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Icon',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: const SafeArea(
          child: Center(
        child: Column(mainAxisAlignment: MainAxisAlignment.center,
children: [
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
            children: [
              Icon(
                Icons.thumb_up_rounded,
              ),
              SizedBox(
                width: 16.0,
              ),
              Icon(
                Icons.thumb_down_rounded,
              ),
            ],
          ),
          SizedBox(
            height: 16.0,
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Icon(
                Icons.thumb_up_rounded,
                color: Colors.green,
                size: 32.0,
              ),
              SizedBox(
                width: 16.0,
              ),
              Icon(
                Icons.thumb_down_rounded,
                color: Colors.red,
                size: 32.0,
              ),
            ],
          ),
        ]),
      )),
    );
  }
}
```

Penjelasan setiap blok pada kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `IconPage`.

4. `class IconPage extends StatefulWidget {…}`

   Halaman ini bersifat `Stateful`, sehingga bisa berubah ketika ada interaksi. Halaman ini akan menampilkan beberapa ikon.

5. `class _IconPageState extends State<IconPage> {`

   Bagian ini mendefinisikan state dari halaman `IconPage`.

6. `Widget build(BuildContext context) {…}`

   Metode ini membangun antarmuka pengguna untuk halaman `IconPage`.

   - `Scaffold`: Menyediakan struktur dasar halaman, termasuk:
     - `AppBar`: Menampilkan bar di bagian atas dengan teks "Icon".
     - `SafeArea`: Menjaga agar konten tidak berada di area yang berisiko tertutup (seperti status bar atau notch).
     - `Center`: Menyelaraskan konten ke tengah layar.
     - `Column`: Menampilkan widget secara vertikal (kolom).
       - `Row`: Mengatur ikon secara horizontal (baris).
         - `Icon(Icons.thumb_up_rounded)`: Menampilkan ikon jempol ke atas.

        - `SizedBox(width: 16.0)`: Memberikan jarak horizontal 16 piksel antar ikon.
        - `Icon(Icons.thumb_down_rounded)`: Menampilkan ikon jempol ke bawah.
    - `SizedBox(height: 16.0)`: Memberikan jarak vertikal antara baris pertama dan kedua.
    - `Icon(Icons.thumb_up_rounded, color: Colors.green, size: 32.0)`: Ikon jempol ke atas dengan warna hijau dan ukuran 32 piksel.
    - `Icon(Icons.thumb_down_rounded, color: Colors.red, size: 32.0)`: Ikon jempol ke bawah dengan warna merah dan ukuran 32 piksel.

Hasil running dari kode program:

BAB II
WIDGET LISTVIEW

2.1. Dasar Teori

ListView adalah widget di Flutter yang digunkana untuk menampilkan daftar widget yang dapat digulir. ListView adalah salah satu widget yang paling umum digunakan untuk menampilkan data dalam bentuk daftar.

2.1.1. ListView.builder

ListView.builder adalah konstruktor dalam Flutter yang digunakan untuk membuat atau menampilkan data panjang atau dinamis secara efisien. ListView.builder memuat elemen-elemen pada tampilan sesuai kebutuhan atau yang dikenal dengan istilah "*lazy loading*." Hal ini memungkinkan Flutter hanya merender item yang ditampilkan di layar, sehingga sangat optimal ketika jumlah elemen sangat besar. ListView.builder membutuhkan dua parameter utama, yaitu `itemCount` untuk menentukan jumlah item, dan `itemBuilder` yang merupakan fungsi untuk membangun tampilan tiap item.

2.1.2. ListView.separated

ListView.separated adalah jenis ListView yang serupa dengan ListView.builder, tetapi menambahkan widget pemisah di antara item-itemnya. ListView.separated digunakan untuk membuat daftar dengan pemisah di antara item. Ini berguna jika ingin menambahkan garis pemisah atau elemen lain di antara item dalam daftar. Pemisah ini bisa berupa garis, jarak, atau widget apa pun yang diinginkan. Fungsinya sangat berguna untuk memperjelas batas antara satu item dengan yang lain. ListView.separated menerima dua fungsi utama, yaitu `itemBuilder` untuk membangun setiap item, dan `separatorBuilder` untuk membangun elemen pemisah antara item-item tersebut.

2.1.3. ListView.custom

ListView.custom memberikan fleksibilitas penuh untuk membuat tampilan list dengan menggunakan delegate yang didefinisikan secara custom. Penggunaan delegate memungkinkan kontrol terhadap bagaimana item-item di dalam ListView dirender dan diposisikan. ListView.custom memberi kontrol penuh atas bagaimana daftar dibangun, memungkinkan untuk menyesuaikan logika pembuatan item dan manajemen scroll. ListView.custom sangat cocok digunakan ketika kontrol penuh atas proses rendering item diperlukan, misalnya dalam kasus di mana performa atau penyesuaian tampilan yang lebih kompleks dibutuhkan.

2.1.4. ListView (default)

ListView (default) adalah bentuk dasar dari ListView yang digunakan untuk menampilkan daftar widget secara vertikal. ListView ini digunakan untuk membuat daftar dengan jumlah item yang sudah diketahui dan relatif kecil. Jika daftar tidak terlalu panjang, penggunaan ListView default adalah cara yang sederhana dan efisien untuk menampilkan elemen-elemen di dalam aplikasi. Pengguna dapat menambahkan item secara langsung sebagai child dari ListView. Namun, jika jumlah elemen besar, ini bisa menyebabkan masalah performa karena seluruh elemen akan dirender sekaligus.

2.2. Praktikum

2.2.1. ListView.builder

Berikut ini adalah kode program penggunaan ListView.builder:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ListviewBuilderPage(),
    );
  }
}

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

  @override
  State<ListviewBuilderPage> createState() => _ListViewBuilderPageState();
}

class _ListViewBuilderPageState extends State<ListviewBuilderPage> {
  List<String> listData = List<String>.generate(100, (index) => 'data
$index');
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'ListView Builder',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: ListView.builder(
        itemCount: listData.length,
        shrinkWrap: true,
        physics: const BouncingScrollPhysics(),
        padding: EdgeInsets.zero,
        itemBuilder: (context, index) => ListTile(
          dense: true,
          title: Text(
            listData[index],
            style: const TextStyle(fontSize: 16.0),
          ),
        ),
      )),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

- `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
- `title`: Memberikan judul aplikasi.
- `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
- `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ListviewBuilderPage`.

4. `class ListviewBuilderPage extends StatefulWidget {…}`
   Halaman ini bersifat `Stateful`, sehingga dapat merespons interaksi pengguna. Halaman ini akan menampilkan daftar menggunakan `ListView.builder`.

5. `class _ListViewBuilderPageState extends State<ListviewBuilderPage> {…}`
   Bagian ini mendefinisikan state dari halaman `ListviewBuilderPage`.

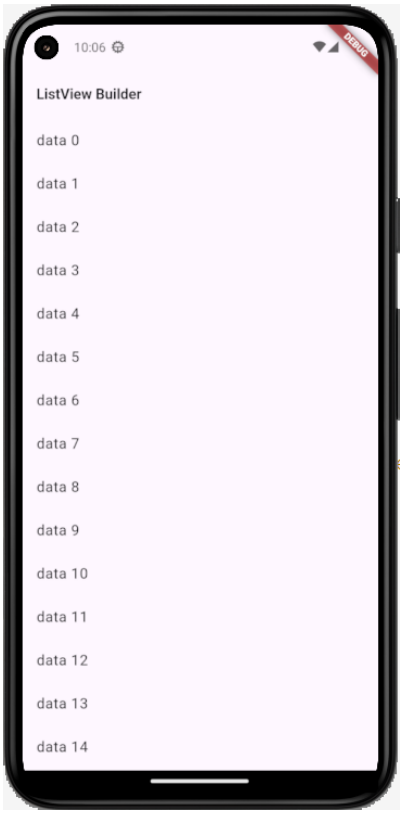
6. `List<String> listData = List<String>.generate(100, (index) => 'data $index');`
   Membuat daftar string yang berisi 100 elemen dengan nama "data 0", "data 1", ..., "data 99".

7. `Widget build(BuildContext context) {…}`
   Metode ini membangun antarmuka pengguna untuk halaman `ListviewBuilderPage`.

   - **`Scaffold`**: Menyediakan struktur dasar halaman, termasuk:
     - **`AppBar`**: Menampilkan bar di bagian atas dengan teks "ListView Builder".
     - **`SafeArea`**: Menghindari elemen UI agar tidak tertutup oleh area sensitif seperti notch atau status bar.
     - **`ListView.builder`**: Widget untuk membuat daftar yang dapat digulir secara efisien:
       - **`itemCount`**: Jumlah total item yang akan ditampilkan, dalam hal ini adalah panjang dari `listData`.
       - **`shrinkWrap`**: Mengatur agar daftar tidak mengambil ruang lebih dari yang diperlukan.
       - **`physics`**: Mengatur efek gulir, dalam hal ini menggunakan `BouncingScrollPhysics` untuk efek gulir elastis.
       - **`padding`**: Mengatur padding untuk daftar. Di sini diatur ke `EdgeInsets.zero` yang berarti tidak ada padding.

- **`itemBuilder`**: Fungsi yang membangun item berdasarkan indeksnya, di sini digunakan untuk membuat **`ListTile`**:
  - **`dense`**: Mengatur agar daftar lebih padat.
  - **`title`**: Menampilkan teks dengan isi dari `listData[index]`, diatur dengan ukuran font 16.0.

Hasil running dari kode program:



2.2.2. ListView.separated

Berikut ini adalah kode program penggunaan ListView.separated:

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


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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
```

```
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ListviewSeparatedPage(),
    );
  }
}

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

  @override
  State<ListviewSeparatedPage> createState() =>
_ListviewSeparatedPageState();
}

class _ListviewSeparatedPageState extends State<ListviewSeparatedPage> {
  List<String> listData = List<String>.generate(100, (index) => 'data
$index');

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'ListView Separated',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: ListView.separated(
        itemCount: listData.length,
        shrinkWrap: true,
        physics: const BouncingScrollPhysics(),
        padding: EdgeInsets.zero,
        itemBuilder: (context, index) => ListTile(
          dense: true,
          title: Text(
            listData[index],
            style: const TextStyle(fontSize: 16.0),
          ),
        ),
        separatorBuilder: (context, index) => const Divider(
          height: 0.0,
        ),
      )),
    );
```

```
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {…}`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ListviewSeparatedPage`.

4. `class ListviewSeparatedPage extends StatefulWidget {…}`

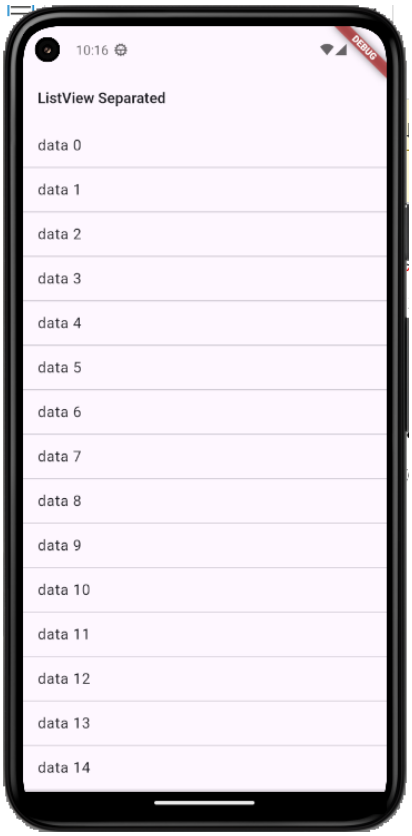
   Widget yang dapat memiliki state yang dapat berubah.

5. `class _ListviewSeparatedPageState extends State<ListviewSeparatedPage> {…}`

   - List<String> listData: Membuat daftar string yang berisi 100 item dengan format "data 0", "data 1", ..., "data 99".
   - Scaffold: Menyediakan struktur dasar halaman, termasuk:
     - AppBar: Menampilkan bar di bagian atas dengan judul "ListView Separated".

- ListView.separated: Widget untuk membuat daftar dengan pemisah antara item:
    - itemCount: Jumlah item yang ditampilkan, diambil dari panjang `listData`.
    - shrinkWrap: Mengatur agar daftar tidak mengambil lebih banyak ruang daripada yang diperlukan.
    - physics: Mengatur efek gulir, menggunakan `BouncingScrollPhysics` untuk efek gulir elastis.
    - padding: Diatur ke `EdgeInsets.zero`, berarti tidak ada padding tambahan.
    - itemBuilder: Fungsi untuk membangun setiap item dalam daftar:
        - Menggunakan ListTile untuk menampilkan teks dari `listData[index]`.
    - separatorBuilder: Fungsi untuk membangun pemisah antara item:
        - Di sini menggunakan Divider dengan tinggi nol, yang berarti pemisah akan terlihat tipis di antara item.

Hasil running dari kode program:



2.2.3. ListView.custom

Berikut ini adalah kode program penggunaan ListView.custom:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ListviewCustomPage(),
    );
  }
}

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

  @override
  State<ListviewCustomPage> createState() => _ListViewCustomPageState();
}

class _ListViewCustomPageState extends State<ListviewCustomPage> {
  List<String> listData = List<String>.generate(100, (index) => 'Data
$index');
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'ListView Custom',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: ListView.custom(
        shrinkWrap: true,
        physics: const BouncingScrollPhysics(),
```

```
        padding: EdgeInsets.zero,
        childrenDelegate: SliverChildBuilderDelegate(
          (context, index) => ListTile(
            dense: true,
            title: Text(
              listData[index],
              style: const TextStyle(fontSize: 16.0),
            ),
          ),
          childCount: listData.length,
        ),
      )),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {…}`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ListviewCustomPage`.

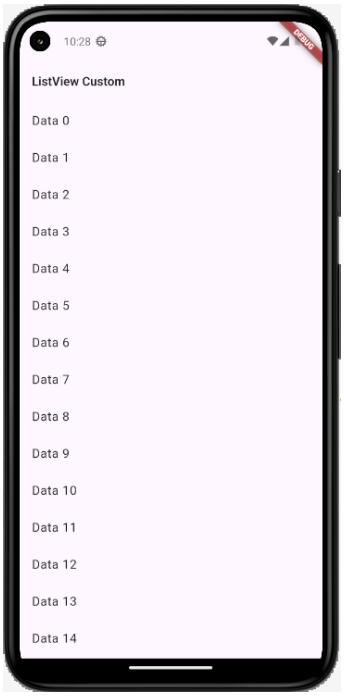
4. `class ListviewCustomPage extends StatefulWidget {…}`

Widget stateful yang memungkinkan konten dinamis ditampilkan.

5. `class _ListViewCustomPageState extends State<ListviewCustomPage> {…}`

- listData: Sebuah daftar string yang dihasilkan menggunakan `List.generate()`, membuat 100 item yang diberi label "Data 0" hingga "Data 99".
- Scaffold: Menyediakan struktur dasar tampilan desain material. Di dalam scaffold:
  - AppBar: Menampilkan judul "ListView Custom".
- ListView.custom: Widget ini menyediakan lebih banyak opsi kustomisasi dibandingkan `ListView` biasa.
  - shrinkWrap: Ketika diatur ke `true`, ini memungkinkan daftar hanya mengambil ruang yang dibutuhkan.
  - physics: Menggunakan `BouncingScrollPhysics` untuk efek pantulan saat menggulir ke atas atau ke bawah daftar.
  - padding: Tidak ada padding tambahan yang diterapkan.
  - childrenDelegate:
    - Menggunakan `SliverChildBuilderDelegate`, yang membangun anak-anak secara malas (lazy):
      - (context, index): Fungsi pembangun yang membuat `ListTile` untuk setiap item dalam `listData`.
      - childCount: Menentukan jumlah anak yang akan dibangun.

Hasil running dari kode program:



2.2.4. ListView (default)

Berikut ini adalah kode program penggunaan ListView (default):

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


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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ListviewPage(),
    );
  }
}

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

  @override
  State<ListviewPage> createState() => _ListViewPageState();
}

class _ListViewPageState extends State<ListviewPage> {
  bool switchEnable = true;

  void onChangeSwitch({required bool value}) {
    setState(() {
      switchEnable = value;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'ListView',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
```

```
      ),
    ),
    body: SafeArea(
        child: ListView(
      shrinkWrap: true,
      physics: const BouncingScrollPhysics(),
      padding: EdgeInsets.zero,
      children: [
        ListTile(
          leading: const Icon(
            Icons.airplanemode_active_rounded,
          ),
          title: const Text(
            'Airplane Mode',
            style: TextStyle(fontSize: 16.0),
          ),
          trailing: Transform.translate(
            offset: const Offset(10.0, 0.0),
            child: Switch(
              value: switchEnable,
              onChanged: (value) => onChangeSwitch(value: value),
            ),
          ),
        ),
        const ListTile(
          leading: Icon(
            Icons.wifi_rounded,
          ),
          title: Text(
            'Wi-Fi',
            style: TextStyle(fontSize: 16.0),
          ),
          trailing: Icon(
            Icons.chevron_right_rounded,
          ),
        ),
        const ListTile(
          leading: Icon(
            Icons.bluetooth_rounded,
          ),
          title: Text(
            'Bluetooth',
            style: TextStyle(fontSize: 16.0),
          ),
          trailing: Icon(Icons.chevron_right_outlined),
        ),
      ],
    )),
```

```
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {…}`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `ListviewPage`.

4. `class ListviewPage extends StatefulWidget {…}`

   Widget stateful yang memungkinkan konten dinamis ditampilkan.

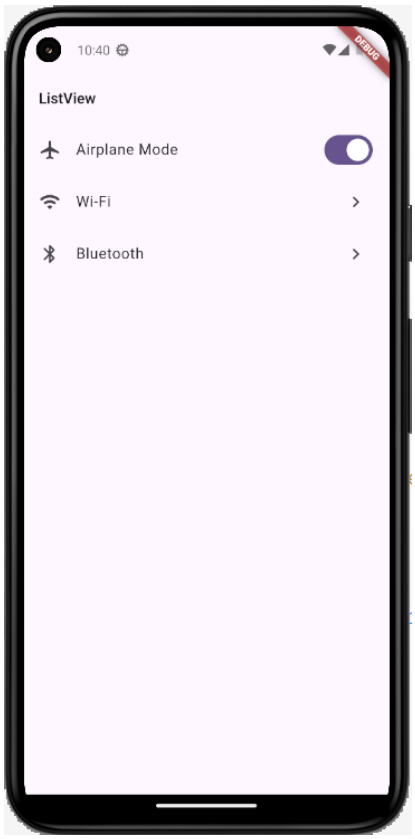
5. `class _ListViewPageState extends State<ListviewPage> {…}`

   - switchEnable: Sebuah boolean yang mengatur status dari switch (on/off). Nilai awalnya adalah `true`.
   - onChangeSwitch: Fungsi ini memanggil `setState()` untuk memperbarui nilai `switchEnable` saat switch berubah, yang kemudian akan memperbarui tampilan.
   - Scaffold: Menyediakan struktur dasar tampilan desain material. Di dalam scaffold:

- AppBar: Menampilkan judul "ListView".

- ListView: Widget ini digunakan untuk menampilkan daftar elemen yang dapat digulir.
  - shrinkWrap: Ketika diatur ke `true`, ini memungkinkan daftar hanya mengambil ruang yang dibutuhkan.
  - physics: Menggunakan `BouncingScrollPhysics` untuk efek pantulan saat menggulir ke atas atau ke bawah daftar.
  - padding: Tidak ada padding tambahan yang diterapkan.
- ListTile: Merupakan widget yang digunakan untuk menampilkan item dalam daftar. Di dalam `ListView`, terdapat beberapa `ListTile`:
  1. ListTile Pertama: Menampilkan ikon pesawat terbang dan judul "Airplane Mode", serta sebuah switch untuk mengaktifkan atau menonaktifkan mode pesawat.
  2. ListTile Kedua: Menampilkan ikon Wi-Fi dan judul "Wi-Fi", dengan ikon panah kanan di sisi kanan.
  3. ListTile Ketiga: Menampilkan ikon Bluetooth dan judul "Bluetooth", juga dengan ikon panah kanan di sisi kanan.

Hasil running dari kode program:



BAB III
WIDGET GRIDVIEW

3.1. Dasar Teori

GridView adalah widget di Flutter yang digunakan untuk menampilkan daftar item dalam bentuk *grid*. GridView sangat berguna untuk membuat *layout* seperti galeri foto atau tampilan produk dalam *e-commerce.*

3.1.1. GridView.count

GridView.count digunakan untuk membuat grid dengan jumlah kolom tetap. Dengan menggunakan properti `crossAxisCount`, pengguna dapat menentukan jumlah kolom yang ingin ditampilkan dalam grid. Ini adalah metode yang sederhana dan efisien untuk menampilkan elemen dalam bentuk grid, terutama ketika jumlah kolom diinginkan untuk tetap konstan.

3.1.2. GridView.extent

GridView.extent digunakan untuk membuat grid dengan lebar kolom tetap. Dalam hal ini, pengguna dapat menentukan lebar maksimal dari kolom dengan properti `maxCrossAxisExtent`. Metode ini memungkinkan elemen dalam grid menyesuaikan diri dengan lebar kolom yang telah ditentukan, sehingga memberikan fleksibilitas dalam pengaturan tampilan grid, terutama ketika jumlah item bervariasi.

3.1.3. GridView.builder

GridView.builder digunakan untuk membuat grid yang panjang atau dinamis. Ini sangat efisien karena hanya widget yang terlihat di layar yang akan dibangun. Dengan cara ini, performa aplikasi dapat lebih baik, terutama ketika jumlah item dalam grid sangat besar. GridView.builder menerima parameter `itemCount` untuk menentukan jumlah item dan `itemBuilder` yang digunakan untuk membangun tampilan setiap item dalam grid.

3.1.4. GridView.custom

GridView.custom memberikan kontrol penuh atas bagaimana grid dibangun, memungkinkan untuk menyesuaikan logika pembuatan item dan manajemen scroll. Dengan menggunakan *delegate* yang didefinisikan secara custom, pengguna dapat mengontrol cara elemen dirender dan diposisikan, yang sangat berguna ketika ada kebutuhan khusus dalam pengaturan grid yang lebih kompleks.

3.2. Praktikum

3.2.1. GridView.count

Berikut ini adalah kode program penggunaan GridView.count:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const GridViewCountPage(),
    );
  }
}

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

  @override
  State<GridViewCountPage> createState() => _GridViewCountPageState();
}

class _GridViewCountPageState extends State<GridViewCountPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'GridView Count',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: GridView.count(
          crossAxisCount: 3,
          mainAxisSpacing: 12.0,
          crossAxisSpacing: 12.0,
```

```
          shrinkWrap: true,
          physics: const BouncingScrollPhysics(),
          padding: const EdgeInsets.all(16.0),
          children: List.generate(
            50,
            (index) => Container(
              decoration: BoxDecoration(
                color: Colors.pink,
                borderRadius: BorderRadius.circular(8.0),
              ),
              child: Center(
                child: Text(
                  'Product $index',
                  style: const TextStyle(fontSize: 16.0, color:
Colors.white),
                  textAlign: TextAlign.center,
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {…}`

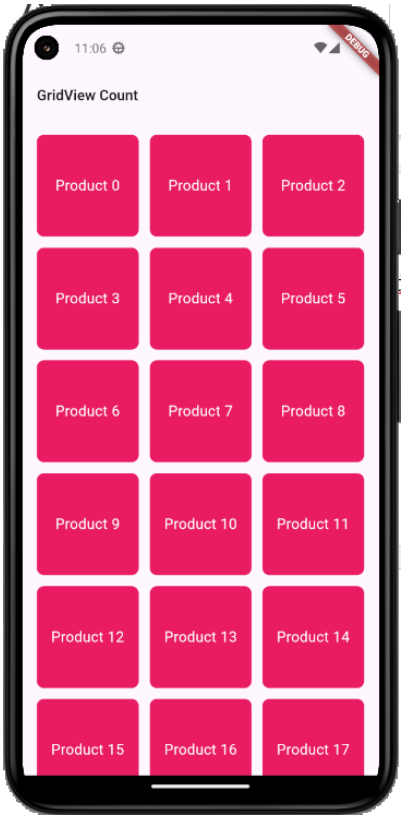
   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.

- `title`: Memberikan judul aplikasi.
- `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
- `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `GridViewCountPage`.

4. `class GridViewCountPage extends StatefulWidget {…}`
   Widget stateful yang memungkinkan konten dinamis ditampilkan.
5. `class _GridViewCountPageState extends State<GridViewCountPage> {…}`
   - Scaffold: Menyediakan struktur dasar tampilan desain material. Di dalam scaffold:
     - AppBar: Menampilkan judul "GridView Count".
   - GridView.count: Widget ini digunakan untuk menampilkan elemen dalam format grid.
     - crossAxisCount: Mengatur jumlah kolom dalam grid, dalam hal ini adalah 3 kolom.
     - mainAxisSpacing: Menentukan jarak vertikal antar item dalam grid, diatur ke 12.0.
     - crossAxisSpacing: Menentukan jarak horizontal antar item dalam grid, diatur ke 12.0.
     - shrinkWrap: Ketika diatur ke `true`, ini memungkinkan grid hanya mengambil ruang yang dibutuhkan.
     - physics: Menggunakan `BouncingScrollPhysics` untuk efek pantulan saat menggulir.
     - padding: Menambahkan padding di sekitar grid sebesar 16.0.
   - List.generate: Membuat daftar 50 item.
     - Container: Setiap item dalam grid adalah sebuah container yang memiliki:
       - BoxDecoration: Mengatur warna latar belakang menjadi merah muda dan memberikan sudut yang membulat (border radius) sebesar 8.0.
       - Center: Mengatur konten di dalam container agar berada di tengah.
       - Text: Menampilkan teks yang menunjukkan produk dengan format 'Product $index', di mana `$index` adalah indeks item.

Hasil running kode program:



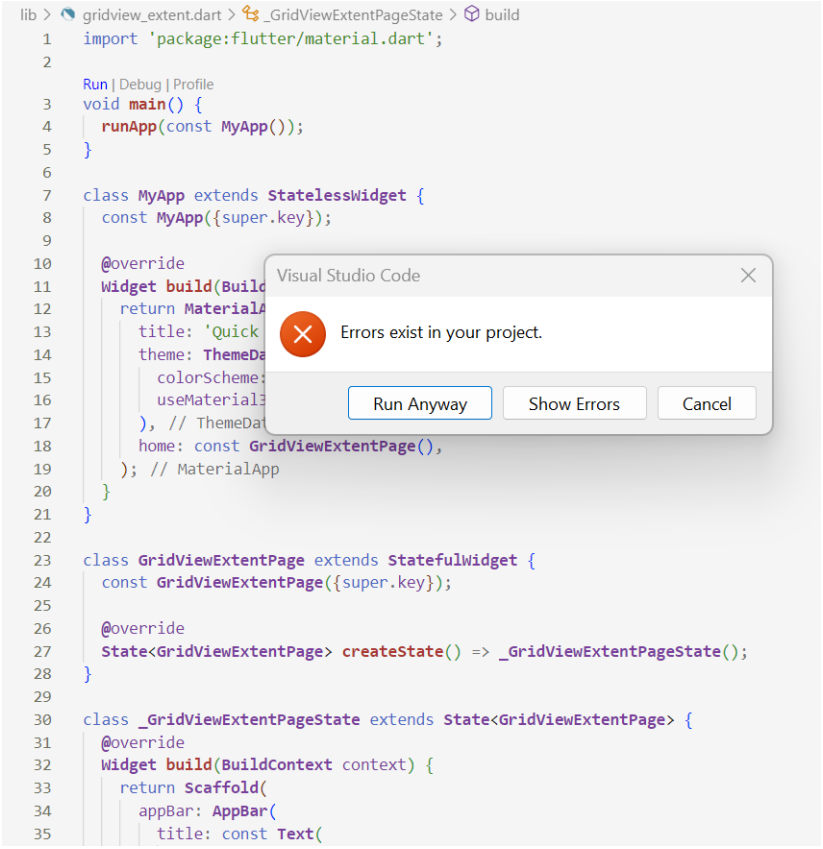
3.2.2. GridView.extent

Berikut ini adalah kode program penggunaan GridView.extent:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const GridViewExtentPage(),
    );
  }
}
```

```dart
class GridViewExtentPage extends StatefulWidget {
  const GridViewExtentPage({super.key});

  @override
  State<GridViewExtentPage> createState() => _GridViewExtentPageState();
}

class _GridViewExtentPageState extends State<GridViewExtentPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'GridView Extent',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: GridView.extent(
          maxCrossAxisExtent: 150.0,
          mainAxisSpacing: 12.0,
          crossAxisSpacing: 12.0,
          shrinkWrap: true,
          physics: const BouncingScrollPhysics(),
          padding: const EdgeInsets.all(16.0),
          children: List.generate(
            50,
            (index) => Container(
              decoration: BoxDecoration(
                color: Colors.blue,
                borderRadius: BorderRadius.circular(8.0),
              ),
              child: Center(
                child: Text(
                  'Product $index',
                  style: const TextStyle(fontSize: 16.0, color:
Colors.white),
                  textAlign: TextAlign.centerr,
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Hasil running dari kode program:



Menemukan kode program yang error:

```
child: Center(
  child: Text(
    'Product $index',
    style: const TextStyle(fontSize: 16.0, color: Colors.white),
    textAlign: TextAlign.centerr,
  ),
```

`TextAlign.centerr` seharusnya ditulis sebagai `TextAlign.center`. Kesalahan ini terjadi karena ada tambahan huruf "r" di akhir kata "center".

Memperbaiki kesalahan:

Dengan memperbaiki ejaan dari `TextAlign.centerr` menjadi `TextAlign.center`.

```
child: Center(
  child: Text(
    'Product $index',
    style: const TextStyle(fontSize: 16.0, color: Colors.white),
    textAlign: TextAlign.center,
  ),
```

Penjelasan setiap blok dari kode program di atas:

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

Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {…}`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `GridViewExtentPage`.

4. `class GridViewExtentPage extends StatefulWidget {…}`

   Widget ini memiliki status (state) yang bisa berubah.

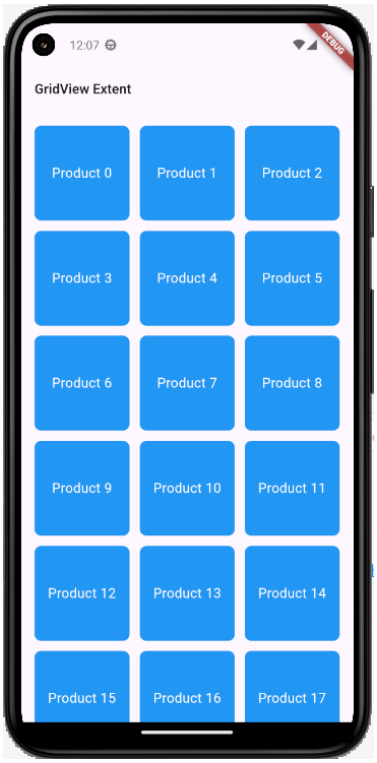
5. `class _GridViewExtentPageState extends State<GridViewExtentPage> {…}`

   - Scaffold: Menyediakan struktur dasar tampilan desain material. Di dalam `Scaffold`:
     - AppBar: Menampilkan judul "GridView Extent".
     - SafeArea: Menghindari konten tertutup oleh elemen antarmuka seperti notifikasi atau status bar.
   - GridView.extent: Widget ini digunakan untuk menampilkan elemen dalam format grid, dengan pengaturan maksimal lebar (cross axis) tiap item:
     - maxCrossAxisExtent: Menentukan lebar maksimal tiap item grid, diatur ke 150.0 piksel.
     - mainAxisSpacing: Menentukan jarak vertikal antar item dalam grid, diatur ke 12.0.

- crossAxisSpacing: Menentukan jarak horizontal antar item dalam grid, diatur ke 12.0.
- shrinkWrap: Ketika diatur ke `true`, ini memungkinkan grid hanya mengambil ruang yang dibutuhkan.
- physics: Menggunakan `BouncingScrollPhysics` untuk efek pantulan saat menggulir.
- padding: Menambahkan padding di sekitar grid sebesar 16.0 piksel.

- List.generate: Membuat daftar 50 item yang masing-masing merupakan `Container`.
- Container: Setiap item dalam grid adalah sebuah `Container` yang memiliki:
  - BoxDecoration: Mengatur warna latar belakang menjadi biru dan memberikan sudut yang membulat (border radius) sebesar 8.0.
  - Center: Mengatur konten di dalam `Container` agar berada di tengah.
  - Text: Menampilkan teks yang menunjukkan produk dengan format 'Product $index', di mana `$index` adalah indeks item dari daftar yang dihasilkan oleh `List.generate`.

Maka, hasil dari kode program setelah diperbaiki:



3.2.3. GridView.builder

Berikut ini adalah kode program penggunaan GridView.builder:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const ListviewBuilderPage(),
    );
  }
}

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

  @override
  State<ListviewBuilderPage> createState() => _ListViewBuilderPageState();
}

class _ListViewBuilderPageState extends State<ListviewBuilderPage> {
  List<String> listData = List<String>.generate(100, (index) => 'data
$index');

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'ListView Builder',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: ListView.builder(
        itemCount: listData.length,
```

```
        shrinkWrap: true,
        physics: const BouncingScrollPhysics(),
        padding: EdgeInsets.zero,
        itemBuilder: (context, index) => ListTile(
          dense: true,
          title: Text(
            listData[index],
            style: const TextStyle(fontSize: 16.0),
          ),
        ),
      )),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {…}`

   Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

   - `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
   - `title`: Memberikan judul aplikasi.
   - `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
   - `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `GridViewBuilderPage`.

4. `class GridviewBuilderPage extends StatefulWidget {…}`

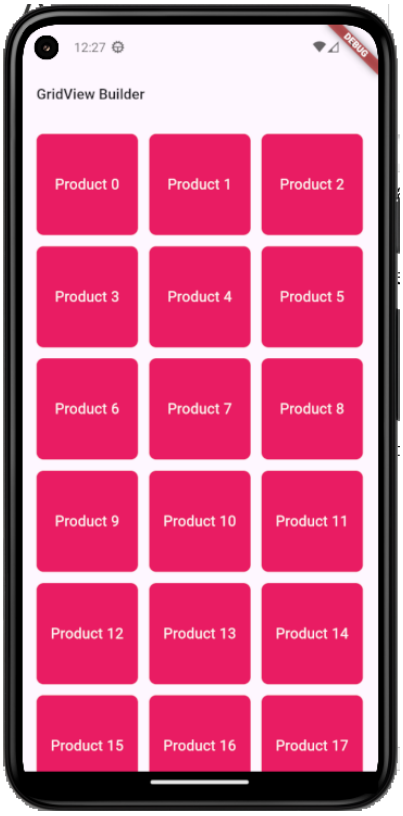
Widget stateful yang memungkinkan konten dinamis ditampilkan. Ini digunakan untuk menampilkan daftar produk dalam format GridView.

5. `class _GridviewBuilderPageState extends State<GridviewBuilderPage> {…}`
List<String> listProduct: Variabel ini menyimpan daftar produk yang terdiri dari 50 item, dengan format "Product $index", di mana $index adalah angka indeks item.

6. `Widget build(BuildContext context) {…}`
- **Scaffold**: Menyediakan struktur dasar tampilan desain material. Di dalam `Scaffold`:
  - **AppBar**: Menampilkan judul "GridView Builder".
- **SafeArea**: Widget ini memastikan konten berada dalam area yang aman, menghindari area yang tertutup oleh status bar, notch, atau elemen UI lain.
- **GridView.builder**: Widget ini digunakan untuk membangun grid yang itemnya dihasilkan secara dinamis:
  - `itemCount`: Menentukan jumlah item yang akan ditampilkan, yaitu panjang dari `listProduct`.
  - `shrinkWrap`: Jika diatur ke `true`, grid hanya akan mengambil ruang yang dibutuhkan.
  - `physics`: Menggunakan **BouncingScrollPhysics** untuk memberikan efek pantulan saat menggulir.
  - `padding`: Menambahkan padding sebesar 16.0 di sekitar grid.
- **SliverGridDelegateWithFixedCrossAxisCount**: Delegasi ini menentukan jumlah kolom dalam grid, serta jarak antar item:
  - `crossAxisCount`: Menentukan jumlah kolom dalam grid, dalam hal ini 3 kolom.
  - `mainAxisSpacing`: Menentukan jarak vertikal antar item, diatur ke 12.0.
  - `crossAxisSpacing`: Menentukan jarak horizontal antar item, diatur ke 12.0.
- **itemBuilder**: Fungsi ini digunakan untuk membangun setiap item dalam grid:
  - **Container**: Setiap item dalam grid adalah sebuah `Container` yang memiliki:

- **BoxDecoration**: Mengatur warna latar belakang menjadi merah muda dan memberikan sudut yang membulat (border radius) sebesar 8.0.
- **Center**: Mengatur konten di dalam container agar berada di tengah.
- **Text**: Menampilkan teks yang menunjukkan produk dengan format `'Product $index'`, di mana `$index` adalah indeks item.

Hasil running dari kode program:



3.2.4. GridView.custom

Berikut ini adalah kode program penggunaan GridView.custom:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
```

```
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const GridviewCustomPage(),
    );
  }
}

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

  @override
  State<GridviewCustomPage> createState() => _GridviewCustomPageState();
}

class _GridviewCustomPageState extends State<GridviewCustomPage> {
  List<String> listProduct =
      List<String>.generate(50, (index) => 'Product $index');

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'GridView Custom',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
          child: GridView.custom(
        shrinkWrap: true,
        physics: const BouncingScrollPhysics(),
        padding: const EdgeInsets.all(16.0),
        gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: 3,
          mainAxisSpacing: 12.0,
          crossAxisSpacing: 12.0,
        ),
        childrenDelegate: SliverChildBuilderDelegate((context, index) {
          if (index == 0) {
            return Container(
              decoration: BoxDecoration(
                color: Colors.red,
                borderRadius: BorderRadius.circular(8.0),
              ),
              child: Center(
                child: Text(
```

```
                  listProduct[index],
                  style: const TextStyle(fontSize: 16.0, color:
Colors.white),
                  textAlign: TextAlign.center,
                ),
              ),
            );
          } else {
            return Container(
              decoration: BoxDecoration(
                color: Colors.blue,
                borderRadius: BorderRadius.circular(8.0),
              ),
              child: Center(
                child: Text(
                  listProduct[index],
                  style: const TextStyle(fontSize: 16.0, color:
Colors.white),
                  textAlign: TextAlign.center,
                ),
              ),
            );
          }
        }, childCount: listProduct.length),
      )),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

2. `void main() { runApp(const MyApp());}`

   Fungsi `main()` adalah titik awal aplikasi Flutter. Fungsi `runApp()` digunakan untuk menjalankan aplikasi dengan widget utama (`MyApp` dalam hal ini).

3. `class MyApp extends StatelessWidget {…}`

   Kelas `MyApp` adalah widget utama aplikasi. Kelas ini adalah turunan dari `StatelessWidget`, yang berarti widget ini bersifat statis (tidak berubah).

   `Widget build(BuildContext context) {…}`

Metode `build()` mendefinisikan bagaimana tampilan aplikasi dibangun. Di sini, ia mengembalikan `MaterialApp`, yang merupakan wadah utama aplikasi yang menggunakan desain Material.

- `MaterialApp`: Komponen utama yang mengelola rute dan tema aplikasi.
- `title`: Memberikan judul aplikasi.
- `theme`: Menerapkan tema dengan menggunakan `ThemeData`. Tema di sini dikustomisasi dengan `ColorScheme.fromSeed()` menggunakan warna dasar `Colors.deepPurple`.
- `home`: Menentukan tampilan utama yang akan ditampilkan, dalam hal ini adalah `GridViewCustomPage`.

4. `class GridviewCustomPage extends StatefulWidget {…}`

Widget stateful yang memungkinkan konten dinamis ditampilkan, seperti grid item yang dapat berubah-ubah.

5. `class _GridviewCustomPageState extends State<GridviewCustomPage> {…}`

List<String> listProduct: Variabel ini menyimpan daftar produk yang terdiri dari 50 item, dengan format "Product $index", di mana $index adalah angka indeks item.

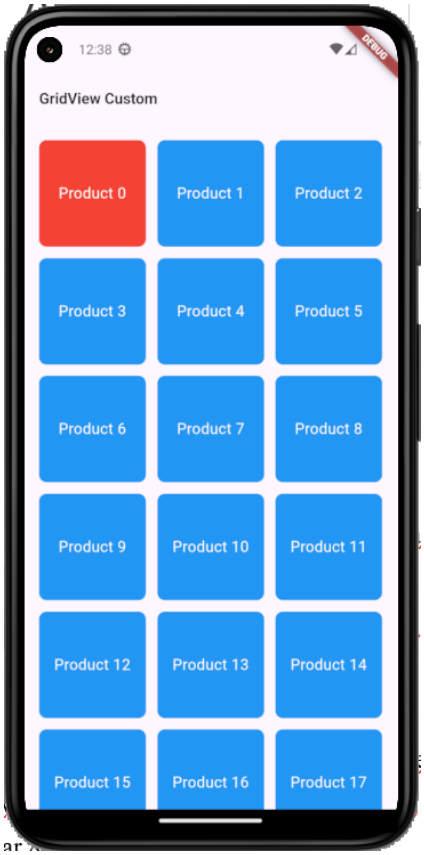
6. `Widget build(BuildContext context) {…}`

- Scaffold: Menyediakan struktur dasar tampilan desain material yang terdiri dari:
  - AppBar: Menampilkan judul "GridView Custom".
- SafeArea: Widget ini memastikan bahwa konten tidak tertutup oleh status bar atau elemen UI lainnya.
- GridView.custom: Widget ini digunakan untuk menampilkan grid secara kustom, dengan item yang dihasilkan dinamis menggunakan `SliverChildBuilderDelegate`. Elemen pentingnya meliputi:
  - shrinkWrap: Mengatur agar grid hanya mengambil ruang yang dibutuhkan.
  - physics: Menggunakan BouncingScrollPhysics untuk memberikan efek pantulan saat menggulir.
  - padding: Menambahkan padding sebesar 16.0 di sekitar grid.
- SliverGridDelegateWithFixedCrossAxisCount: Mengatur grid dengan jumlah kolom tetap:

- `crossAxisCount`: Menentukan jumlah kolom, dalam hal ini adalah 3 kolom.
- `mainAxisSpacing`: Mengatur jarak vertikal antar item.
- `crossAxisSpacing`: Mengatur jarak horizontal antar item.

- SliverChildBuilderDelegate: Delegate ini menentukan bagaimana setiap item di dalam grid dibangun secara dinamis:
  - Jika `index` adalah 0, maka item pertama di dalam grid akan diberi warna latar belakang merah (`Colors.red`).
  - Item-item lainnya diatur dengan warna biru (`Colors.blue`).
  - Setiap item ditampilkan dalam sebuah Container yang memiliki BoxDecoration dengan warna latar belakang dan borderRadius untuk membulatkan sudut sebesar 8.0.

Hasil running dari koe program:



BAB IV
FLUTTER NAVIGATION

4.1. Dasar Teori

4.1.1. Navigator

Navigator adalah objek yang mengelola tumpukan (*stack*) dari objek Route dalam aplikasi Flutter. Navigator memungkinkan aplikasi untuk berpindah antar halaman dengan menambahkan atau menghapus halaman dari tumpukan. Setiap kali melakukan operasi navigasi, seperti berpindah ke halaman lain atau kembali ke halaman sebelumnya, operasi ini terjadi pada tumpukan Navigator. Dengan cara ini, Navigator memberikan kontrol penuh terhadap alur aplikasi, seperti navigasi maju, mundur, atau mengganti halaman secara dinamis.

4.1.2. Named Routes

Named Routes adalah metode untuk mendefinisikan rute dalam aplikasi Flutter dengan memberikan nama pada setiap rute tersebut. Dengan menggunakan named routes, aplikasi dapat dengan mudah merujuk dan mengakses halaman-halaman yang diinginkan melalui nama rute yang telah ditetapkan. Pendekatan ini memudahkan pengelolaan rute, terutama dalam aplikasi yang memiliki banyak halaman, karena developer hanya perlu menggunakan nama rute alih-alih harus mengingat atau memanipulasi objek rute secara langsung.

4.2. Praktikum

4.2.1. Navigator.push & Navigator.pop

Berikut ini adalah kode program penggunaan Navigator.push:

```dart
import 'package:flutter/material.dart';
import 'package:project/detail.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
```

```
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const PushNavigationPage(),
    );
  }
}

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

  @override
  State<PushNavigationPage> createState() => _PushNavigationPageState();
}

class _PushNavigationPageState extends State<PushNavigationPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Push Navigation',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
          child: ElevatedButton(
            onPressed: () => Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => const DetailPage(),
                )),
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              foregroundColor: Colors.white,
            ),
            child: const Text(
              'Go to Detail Page',
            ),
          ),
        ),
      ),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

```
import 'package:project/detail.dart';
```
Baris ini mengimpor paket Material Flutter dan file detail.dart yang berisi halaman detail (kemungkinan berada di dalam folder `project`).

2. `void main() {runApp(const MyApp());}`
Ini adalah titik masuk aplikasi. Fungsi `runApp` digunakan untuk memulai aplikasi dengan widget `MyApp`.

3. `class MyApp extends StatelessWidget {…}`
- StatelessWidget: `MyApp` adalah widget stateless yang berfungsi sebagai struktur utama aplikasi.
- MaterialApp: Widget ini menjadi titik masuk aplikasi yang berbasis desain material. Terdapat beberapa parameter utama:
    - `title`: Menentukan judul aplikasi.
    - `theme`: Menerapkan tema aplikasi menggunakan skema warna yang dibangun dari `Colors.deepPurple`.
    - `home`: Halaman awal ditetapkan ke `PushNavigationPage`.

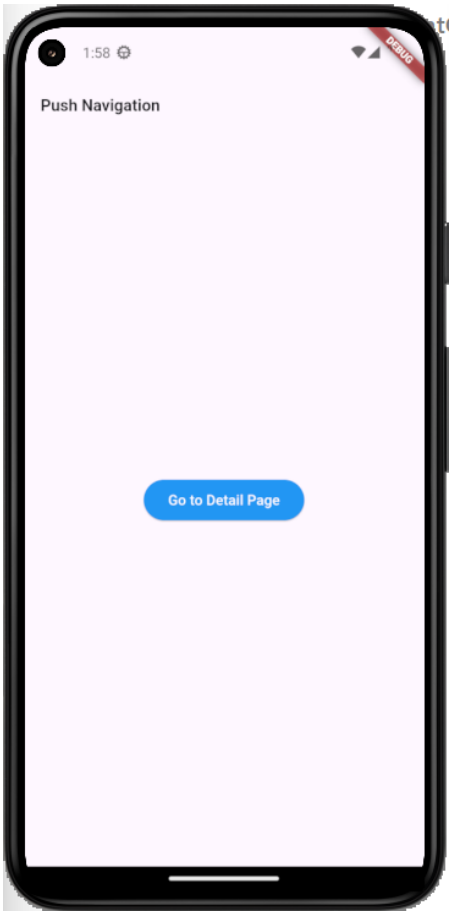
4. `class PushNavigationPage extends StatefulWidget {…}`
Widget yang stateful, artinya memungkinkan perubahan status saat berinteraksi dengan halaman ini.

5. `class _PushNavigationPageState extends State<PushNavigationPage> {…}`
- Scaffold**:** Struktur dasar tampilan untuk halaman ini, menyediakan `AppBar` dan area untuk konten utama.
- AppBar**:** Menampilkan judul "Push Navigation" pada bagian atas halaman.
- SafeArea**:** Memastikan konten aplikasi berada dalam area aman, tidak tertutup status bar atau elemen UI lainnya.
- Center**:** Membuat semua konten di halaman berada di tengah.
- ElevatedButton**:** Tombol yang diangkat, diatur dengan gaya khusus, seperti:
    - `backgroundColor`: Warna latar belakang tombol diatur menjadi biru.
    - `foregroundColor`: Warna teks tombol diatur menjadi putih.
- Navigator.push**:** Fungsi ini digunakan untuk berpindah ke halaman baru. Ketika tombol ditekan, halaman `DetailPage` akan dibuka menggunakan `MaterialPageRoute`, yang mengelola transisi halaman dengan animasi standar.

6. DetailPage (dari file detail.dart)
Ini adalah halaman yang ditampilkan ketika tombol ditekan, dan dituliskan dalam file detail.dart.

Kode ini membuat aplikasi dengan tombol yang memungkinkan pengguna untuk berpindah ke halaman detail menggunakan navigasi `Navigator.push`. Ketika tombol ditekan, halaman detail akan dibuka dengan menggunakan transisi standar dari `MaterialPageRoute`.

Hasil running dari kode program:



4.2.2. Navigator.pushReplacement

Berikut ini adalah kode program penggunaan Navigator.pushReplacement:

```dart
import 'package:flutter/material.dart';
import 'package:project/detail.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
```

```
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const PushreplaceNavigationPage(),
    );
  }
}

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

  @override
  State<PushreplaceNavigationPage> createState() =>
      _PushreplaceNavigationPageState();
}

class _PushreplaceNavigationPageState extends
State<PushreplaceNavigationPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Push Replace Navigation',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
          child: ElevatedButton(
            onPressed: () => Navigator.pushReplacement(
                context,
                MaterialPageRoute(
                  builder: (context) => const DetailPage(),
                )),
            style: ElevatedButton.styleFrom(
              backgroundColor: Colors.blue,
              foregroundColor: Colors.white,
            ),
            child: const Text('Go to Detail Page'),
          ),
        ),
      ),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

1. ```
   import 'package:flutter/material.dart';
   import 'package:project/detail.dart';
   ```
   Baris ini mengimpor paket Material Flutter dan file detail.dart yang berisi halaman detail (kemungkinan berada di dalam folder `project`).

2. ```
   void main() {runApp(const MyApp());}
   ```
   Ini adalah titik masuk aplikasi. Fungsi `runApp` digunakan untuk memulai aplikasi dengan widget `MyApp`.

3. ```
   class MyApp extends StatelessWidget {…}
   ```
   - StatelessWidget: `MyApp` adalah widget stateless yang berfungsi sebagai struktur utama aplikasi.
   - MaterialApp: Widget ini menjadi titik masuk aplikasi yang berbasis desain material. Terdapat beberapa parameter utama:
     - `title`: Menentukan judul aplikasi.
     - `theme`: Menerapkan tema aplikasi menggunakan skema warna yang dibangun dari `Colors.deepPurple`.
     - `home`: Halaman awal ditetapkan ke `PushreplaceNavigationPage`.

4. ```
   class PushreplaceNavigationPage extends StatefulWidget {…}
   ```
   Widget yang bisa berubah saat aplikasi dijalankan. Menyediakan elemen navigasi dengan state dinamis.

5. ```
   class _PushreplaceNavigationPageState extends
   State<PushreplaceNavigationPage> {
   ```
   - Scaffold: Menyediakan struktur dasar untuk halaman ini, termasuk `AppBar` dan body yang berisi konten utama.
   - AppBar: Tampil pada bagian atas halaman dengan judul "Push Replace Navigation".
   - SafeArea: Melindungi elemen UI agar tidak terhalang oleh status bar atau area yang tidak aman.
   - Center: Mengatur posisi widget di tengah layar.
   - ElevatedButton: Membuat tombol berwarna biru dengan teks putih, yang berfungsi untuk mengarahkan ke halaman detail.
     - onPressed: Ketika tombol ditekan, fungsi `Navigator.pushReplacement` digunakan untuk menggantikan halaman saat ini dengan `DetailPage`. Ini berarti halaman sebelumnya tidak disimpan di stack navigator, sehingga pengguna tidak dapat kembali ke halaman ini menggunakan tombol kembali (back button).
   ```

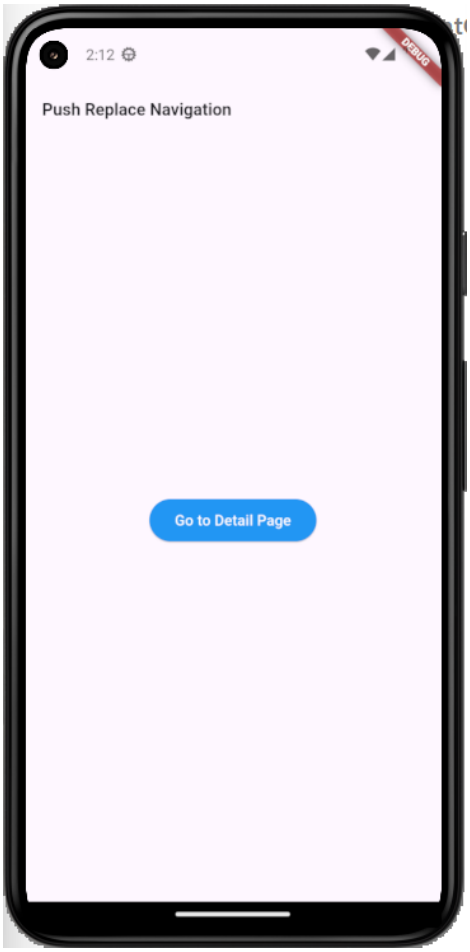
- MaterialPageRoute: Menyediakan animasi transisi standar saat berpindah ke halaman baru, yaitu `DetailPage`.

6. DetailPage (dari file detail.dart)

   Halaman `DetailPage` berada dalam file eksternal detail.dart, dan ditampilkan ketika tombol pada halaman `PushreplaceNavigationPage` ditekan.

Kode ini menunjukkan penggunaan navigasi dengan penggantian halaman menggunakan `Navigator.pushReplacement`. Saat tombol ditekan, pengguna akan diarahkan ke halaman `DetailPage`, dan halaman saat ini (PushreplaceNavigationPage) tidak disimpan di stack, sehingga tidak bisa kembali ke halaman ini menggunakan tombol back.

Hasil running dari kode program:



4.2.3. Navigator.pushAndRemoveUntil

Berikut ini adalah kode program penggunaan Navigator.pushAndRemoveUntil:

```
import 'package:flutter/material.dart';
import 'package:project/detail.dart';

void main() {
```

```
  runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quick Note',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const PushremoveuntilNavigationPage(),
    );
  }
}

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

  @override
  State<PushremoveuntilNavigationPage> createState() =>
      _PushremoveuntilNavigationPageState();
}

class _PushremoveuntilNavigationPageState
    extends State<PushremoveuntilNavigationPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Push Removeuntil Navigation',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
          child: ElevatedButton(
            onPressed: () => Navigator.pushAndRemoveUntil(
              context,
              MaterialPageRoute(
                builder: (context) => const DetailPage(),
              ),
              (route) => false,
            ),
```

```
            style: ElevatedButton.styleFrom(
                backgroundColor: Colors.red, foregroundColor:
Colors.white),
            child: const Text('Go to Detail Page'),
          ),
        ),
      ),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

1. `import 'package:flutter/material.dart';`
   `import 'package:project/detail.dart';`
   Baris ini mengimpor paket Material Flutter dan file detail.dart yang berisi halaman detail (kemungkinan berada di dalam folder `project`).

2. `void main() {runApp(const MyApp());}`
   Ini adalah titik masuk aplikasi. Fungsi `runApp` digunakan untuk memulai aplikasi dengan widget `MyApp`.

3. `class MyApp extends StatelessWidget {…}`
   - StatelessWidget: `MyApp` adalah widget stateless yang berfungsi sebagai struktur utama aplikasi.
   - MaterialApp: Widget ini menjadi titik masuk aplikasi yang berbasis desain material. Terdapat beberapa parameter utama:
     - `title`: Menentukan judul aplikasi.
     - `theme`: Menerapkan tema aplikasi menggunakan skema warna yang dibangun dari `Colors.deepPurple`.
     - `home`: Halaman awal ditetapkan ke `PushremoveuntilNavigationPage`.

4. `class PushremoveuntilNavigationPage extends StatefulWidget {`
   Widget yang memungkinkan perubahan data saat aplikasi berjalan.

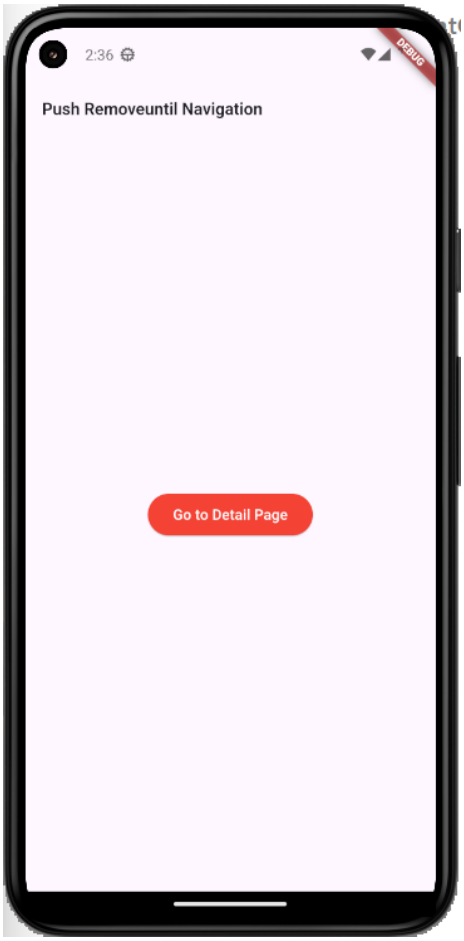
5. `class _PushremoveuntilNavigationPageState extends State<PushremoveuntilNavigationPage> {`
   - Scaffold: Menyediakan struktur dasar halaman ini, termasuk `AppBar` dan body.
   - AppBar: Menampilkan bagian atas halaman dengan judul "Push Removeuntil Navigation".
   - SafeArea: Memastikan konten UI tidak terhalang oleh status bar atau area yang tidak aman.
   - Center: Mengatur posisi widget di tengah layar.

- ElevatedButton: Tombol yang berwarna merah dengan teks putih. Ketika tombol ditekan, pengguna akan diarahkan ke halaman detail menggunakan Navigator.pushAndRemoveUntil.
    - onPressed: Fungsi ini akan menggantikan halaman saat ini dan menghapus semua halaman sebelumnya dari stack navigasi.
    - Navigator.pushAndRemoveUntil:
        - MaterialPageRoute: Membuat rute baru untuk halaman `DetailPage`.
        - (route) => false: Semua halaman sebelumnya akan dihapus, sehingga pengguna tidak bisa kembali ke halaman sebelumnya menggunakan tombol kembali (back button).

6. DetailPage (Dari file detail.dart)

   Halaman `DetailPage` berada dalam file eksternal detail.dart dan ditampilkan ketika tombol ditekan.

Hasil running dari kode program:



4.2.4. Inisialisasi Named Routes

Berikut ini adalah kode program penggunaan Named Routes:

```
import 'package:flutter/material.dart';
import 'package:project/detail.dart';
import 'package:project/home.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        title: 'Quick Note',
        theme: ThemeData(
          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
          useMaterial3: true,
        ),
        initialRoute: '/',
        routes: {
          '/': (context) => const HomePage(),
          '/detail': (context) => const DetailPage()
        });
  }
}
```

Penjelasan setiap blok dari kode program di atas:

1. `import 'package:flutter/material.dart';`
   `import 'package:project/detail.dart';`
   `import 'package:project/home.dart';`
   Baris ini mengimpor paket Material Flutter dan file detail.dart yang berisi halaman detail (kemungkinan berada di dalam folder `project`).

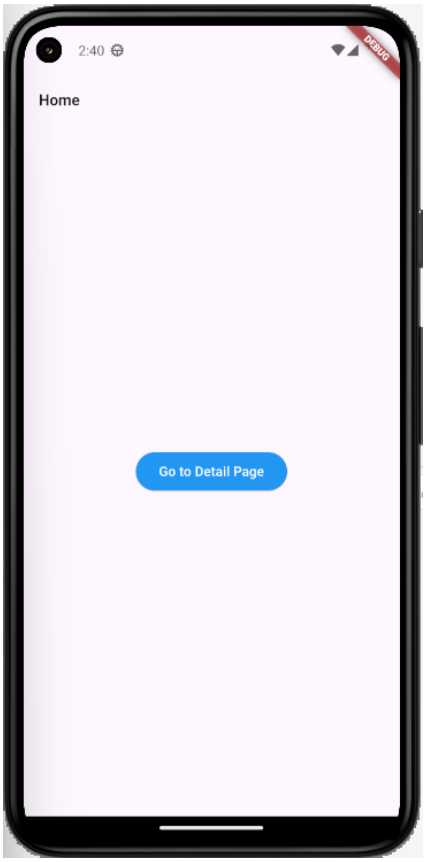
2. `void main() {runApp(const MyApp());}`
   Ini adalah titik masuk aplikasi. Fungsi `runApp` digunakan untuk memulai aplikasi dengan widget `MyApp`.

3. `class MyApp extends StatelessWidget {…}`
   - StatelessWidget: `MyApp` adalah widget stateless yang membangun struktur utama aplikasi.
   - MaterialApp: Widget ini berfungsi sebagai root dari aplikasi dengan beberapa konfigurasi:
     1. title: Judul aplikasi yang akan ditampilkan.

2. theme: Menerapkan tema dengan skema warna yang berasal dari warna dasar `deepPurple` dan menggunakan Material 3.
3. initialRoute: Menetapkan rute awal aplikasi ke `'/'`, yang merujuk pada halaman beranda.
4. routes: Menyediakan pemetaan antara rute dan widget yang akan ditampilkan:
    - `'/': (context) => const HomePage()`: Rute ini merujuk ke halaman beranda.
    - `'/detail': (context) => const DetailPage()`: Rute ini merujuk ke halaman detail.

Program ini menggunakan sistem navigasi berbasis rute. Ketika aplikasi dijalankan, pengguna akan diarahkan ke halaman `HomePage` sebagai halaman awal. Halaman lain, seperti `DetailPage`, dapat diakses melalui rute yang ditentukan. Ini memungkinkan navigasi yang lebih terstruktur dan terorganisir antar halaman dalam aplikasi.

Hasil running dari kode program:



4.2.5. Membuat halaman utama (*HomeScreen*)

Berikut ini adalah kode program untuk membuat halaman utama (*HomeScreen*):

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

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

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

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Home',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
          child: ElevatedButton(
            onPressed: () => Navigator.pushNamed(context, '/detail'),
            style: ElevatedButton.styleFrom(
                backgroundColor: Colors.blue, foregroundColor:
Colors.white),
            child: const Text('Go to Detail Page'),
          ),
        ),
      ),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

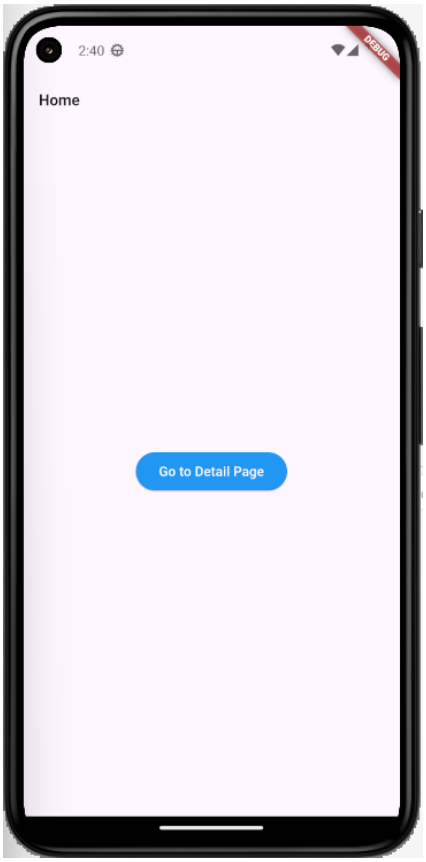
2. `class HomePage extends StatefulWidget {…}`

   Widget stateful, yang berarti ia memiliki state yang dapat berubah seiring waktu.

3. `class _HomePageState extends State<HomePage> {…}`

- State Class: `_HomePageState` adalah implementasi dari state untuk `HomePage`.
- Scaffold: Menyediakan struktur dasar tampilan, termasuk:
    - AppBar: Menampilkan judul "Home" dengan gaya teks yang ditentukan.
    - SafeArea: Menghindari konten yang ditampilkan tertutup oleh notifikasi atau area layar yang terpotong.
    - Center: Menempatkan konten di tengah halaman.
    - ElevatedButton: Tombol yang dapat ditekan untuk navigasi ke halaman detail.
        - onPressed: Ketika tombol ditekan, aplikasi akan mengarahkan pengguna ke halaman detail menggunakan `Navigator.pushNamed(context, '/detail')`.
        - style: Mengatur warna latar belakang tombol menjadi biru dan teks menjadi putih.
        - child: Teks tombol yang menampilkan "Go to Detail Page".

Ini adalah tampilan ketiga `HomePage` diakses:



4.2.6. Membuat halaman detail (*DetailScreen*)

Berikut ini adalah kode program untuk membuat halaman detail (*DetailScreen*):

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

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

  @override
  State<DetailPage> createState() => _DetailPageState();
}

class _DetailPageState extends State<DetailPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Detail',
          style: TextStyle(fontSize: 16.0, fontWeight: FontWeight.w600),
        ),
      ),
      body: SafeArea(
        child: Center(
            child: ElevatedButton(
          onPressed: () => Navigator.pop(context),
          style: ElevatedButton.styleFrom(
              backgroundColor: Colors.red, foregroundColor: Colors.white),
          child: const Text('Go back to Home'),
        )),
      ),
    );
  }
}
```

Penjelasan setiap blok dari kode program di atas:

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

   Kode ini mengimpor Pustaka material design Flutter. Pustaka ini menyediakan berbagain widget yang mengikuti gaya desain Material, seperti button, app bar, dan lain-lain.

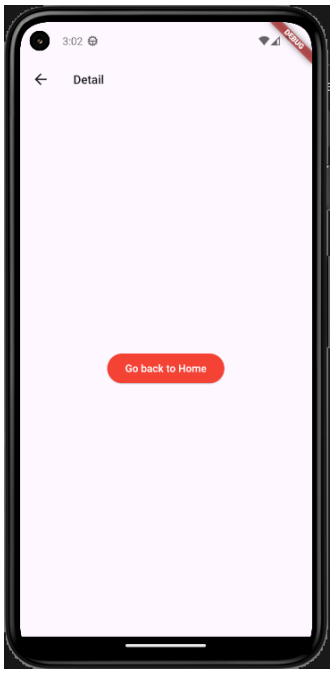
2. `class DetailPage extends StatefulWidget {…}`

   - StatefulWidget: `DetailPage` adalah widget stateful, yang berarti memiliki state yang bisa berubah seiring waktu.
   - createState: Metode ini digunakan untuk membuat instansi dari state yang terkait dengan `DetailPage`, yaitu `_DetailPageState`.

3. `class _DetailPageState extends State<DetailPage> {…}`
    - State Class: `_DetailPageState` adalah implementasi dari state untuk `DetailPage`.
    - Scaffold: Menyediakan struktur dasar tampilan, termasuk:
        - AppBar: Menampilkan judul "Detail" dengan gaya teks yang ditentukan.
        - SafeArea: Menghindari konten yang ditampilkan tertutup oleh notifikasi atau area layar yang terpotong.
        - Center: Menempatkan konten di tengah halaman.
        - ElevatedButton: Tombol yang dapat ditekan untuk kembali ke halaman sebelumnya (halaman utama).
            - onPressed: Ketika tombol ditekan, aplikasi akan kembali ke halaman sebelumnya menggunakan `Navigator.pop(context)`.
            - style: Mengatur warna latar belakang tombol menjadi merah dan teks menjadi putih.
            - child: Teks tombol yang menampilkan "Go back to Home".

Halaman `DetailPage` menyajikan antarmuka yang sederhana dengan sebuah tombol. Ketika tombol tersebut ditekan, pengguna akan diarahkan kembali ke halaman sebelumnya (halaman beranda) menggunakan sistem navigasi `pop`. Ini memberikan pengalaman pengguna yang intuitif dan sederhana dalam menjelajahi aplikasi.

Ini adalah tampilan ketiga `DetailPage` diakses:



DAFTAR PUSTAKA

Anggara, A., & Pratama, D. (2024). Modul Praktikum Mobile & Web Service. Yogyakarta: Universitas Tekonologi Yogyakarta.

Stack class. (n.d.). Diakses dari https://api.flutter.dev/flutter/widgets/Stack-class.html

Padding class. (n.d.). Diakses dari https://api.flutter.dev/flutter/widgets/Padding-class.html

Align class. (n.d.). Diakses dari https://api.flutter.dev/flutter/widgets/Align-class.html

ElevatedButton class. (n.d.). Diakses dari https://api.flutter.dev/flutter/material/ElevatedButton-class.html

TextField class. (n.d.). Diakses dari https://api.flutter.dev/flutter/material/TextField-class.html

Image class. (n.d.). Diakses dari https://api.flutter.dev/flutter/widgets/Image-class.html

Container class. (n.d.). Diakses dari https://api.flutter.dev/flutter/widgets/Container-class.html

Icon class. (n.d.). Diakses dari https://api.flutter.dev/flutter/widgets/Icon-class.html

BuildWithAngga. (n.d.). Mengenal ListView: Widget Untuk Menampilkan Daftar Item Pada Flutter. Diakses dari https://buildwithangga.com/tips/mengenal-listview-widget-untuk-menampilkan-daftar-item-pada-flutter

Setiawan, Y. (2019). Flutter: GridView. Diakses dari https://medium.com/nusanet/flutter-gridview-bad48c1f216c

Bhaumik, P. (2021). Flutter: Push, Pop, Push. Diakses dari https://medium.com/flutter-community/flutter-push-pop-push-1bb718b13c31