

**PEMROGRAMAN MOBILE
PENGANTAR BAHASA PEMROGRAMAN DART -
BAGIAN 7**



OLEH:

Nama : Agung Rizky S

NIM : 2241720187

Kelas : TI – 3C

**PROGRAM STUDI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG**

2024

Praktikum 1: Membangun Layout di Flutter

Langkah 1: Buat Project Baru

Buatlah sebuah project flutter baru dengan nama **layout_flutter**. Atau sesuaikan style laporan praktikum yang Anda buat.

Langkah 2: Buka file lib/main.dart

Buka file `main.dart` lalu ganti dengan kode berikut. Isi nama dan NIM Anda di `text title`.

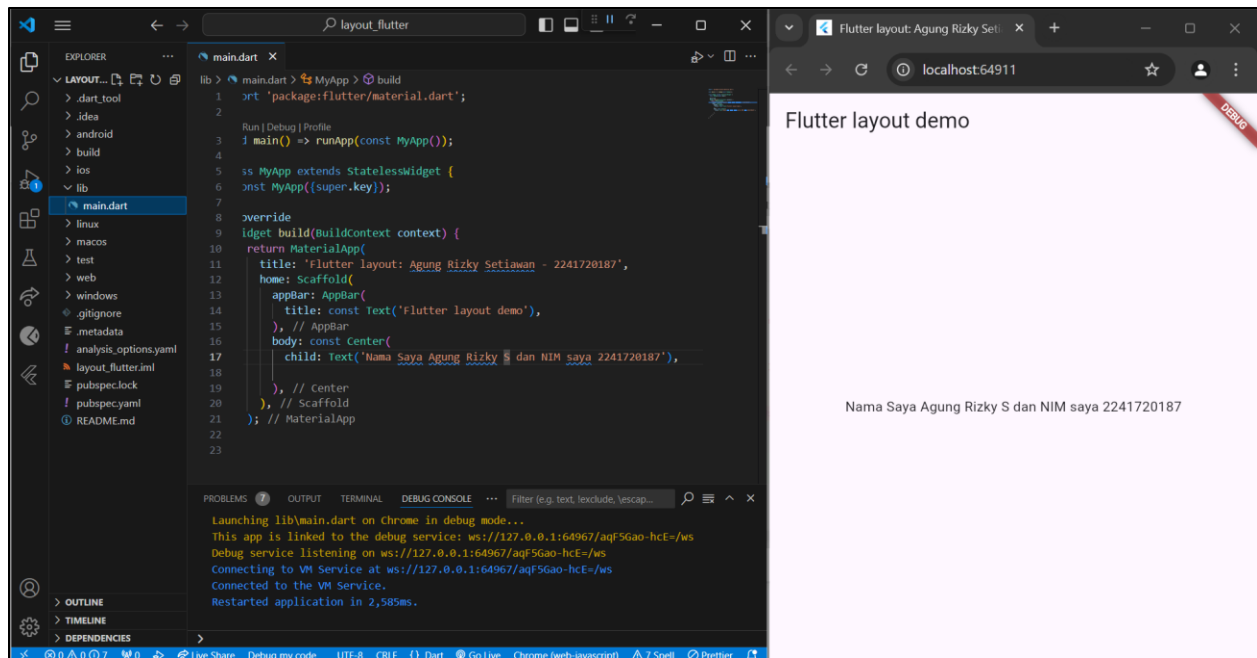
```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout: Nama dan NIM Anda',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Flutter layout demo'),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Jawaban :

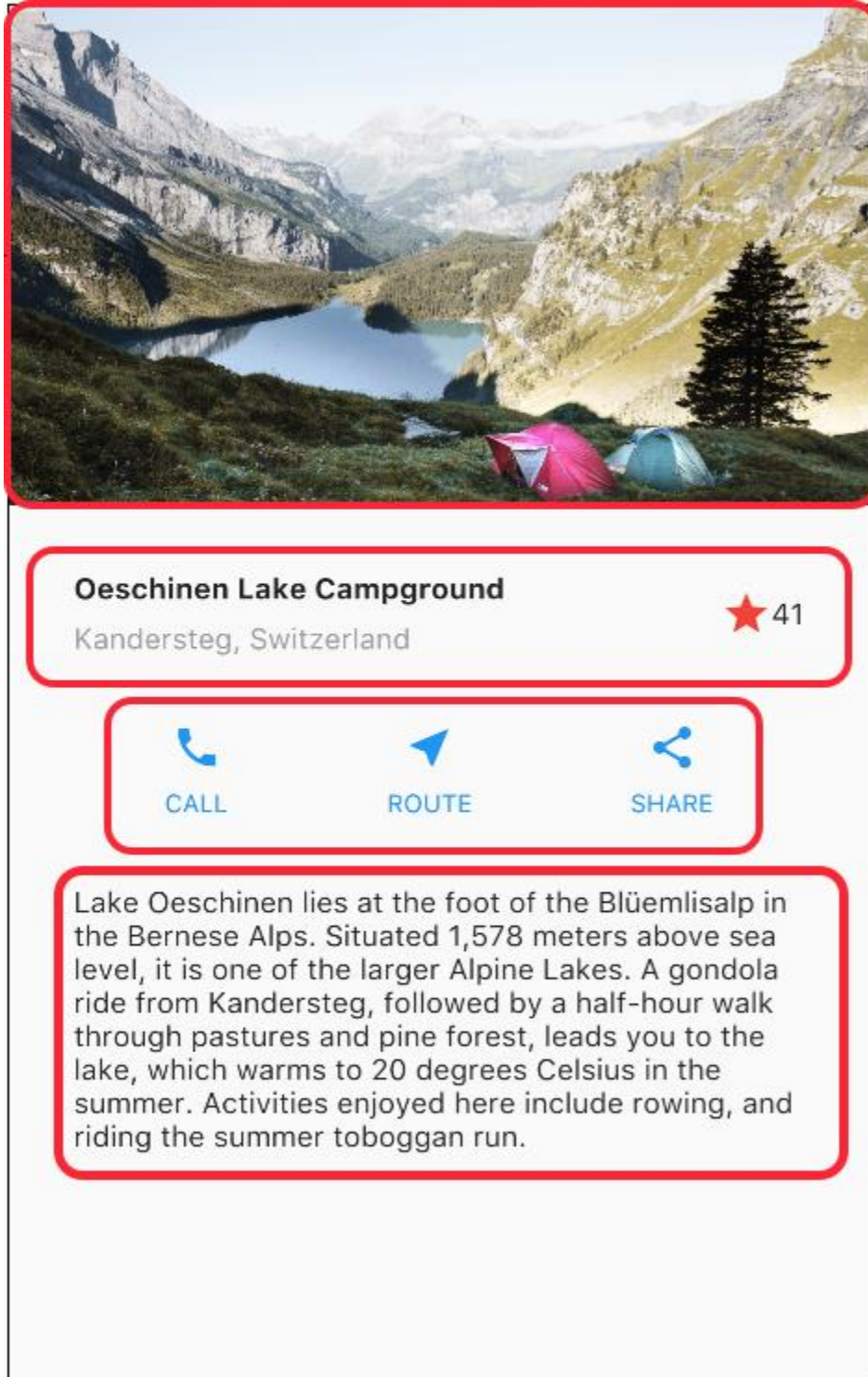


Langkah 3: Identifikasi layout diagram

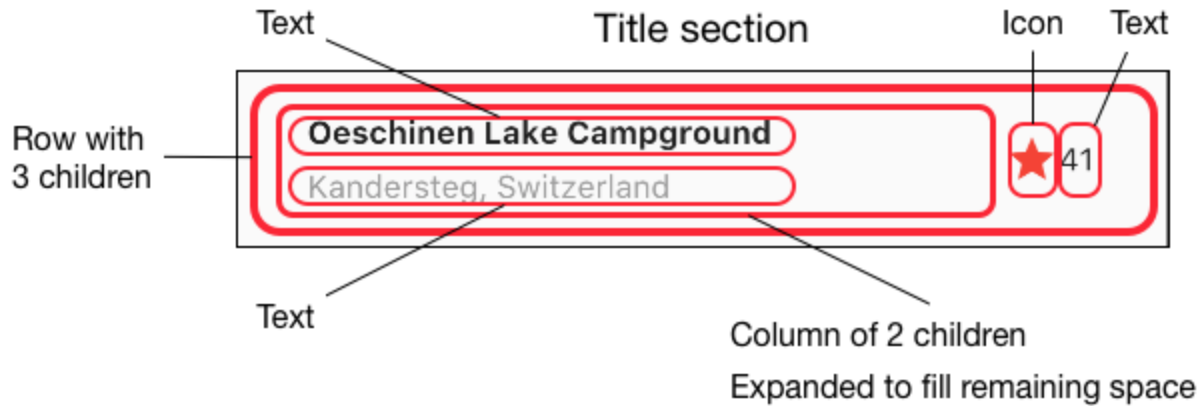
Langkah pertama adalah memecah tata letak menjadi elemen dasarnya:

- Identifikasi baris dan kolom.
- Apakah tata letaknya menyertakan kisi-kisi (grid)?
- Apakah ada elemen yang tumpang tindih?
- Apakah UI memerlukan tab?
- Perhatikan area yang memerlukan alignment, padding, atau borders.

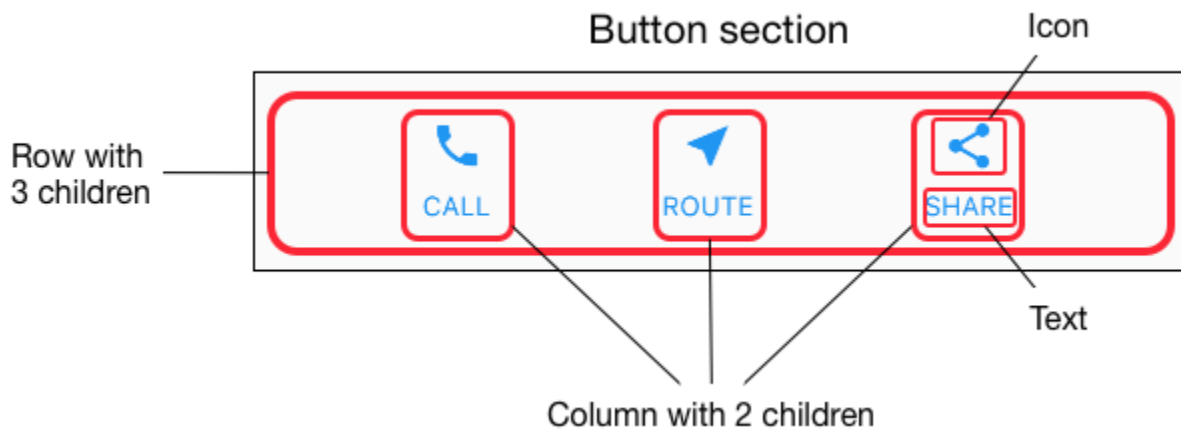
Pertama, identifikasi elemen yang lebih besar. Dalam contoh ini, empat elemen disusun menjadi sebuah kolom: sebuah gambar, dua baris, dan satu blok teks.



Selanjutnya, buat diagram setiap baris. Baris pertama, disebut bagian Judul, memiliki 3 anak: kolom teks, ikon bintang, dan angka. Anak pertamanya, kolom, berisi 2 baris teks. Kolom pertama itu memakan banyak ruang, sehingga harus dibungkus dengan widget yang Diperluas.



Baris kedua, disebut bagian Tombol, juga memiliki 3 anak: setiap anak merupakan kolom yang berisi ikon dan teks.



Setelah tata letak telah dibuat diagramnya, cara termudah adalah dengan menerapkan pendekatan bottom-up. Untuk meminimalkan kebingungan visual dari kode tata letak yang banyak bertumpuk, tempatkan beberapa implementasi dalam variabel dan fungsi.

Langkah 4: Implementasi title row

Pertama, Anda akan membuat kolom bagian kiri pada judul. Tambahkan kode berikut di bagian atas metode `build()` di dalam kelas `MyApp`:


```
Widget titleSection = Container(
  padding: const EdgeInsets.all(...),
  child: Row(
    children: [
      Expanded(
        /* soal 1 */
        child: Column(
          crossAxisAlignment: ...,
          children: [
            /* soal 2 */
            Container(
```

```
padding: const EdgeInsets.only(bottom: ...),
child: const Text(
  'Wisata Gunung di Batu',
  style: TextStyle(
    fontWeight: FontWeight.bold,
  ),
),
),
Text(
  'Batu, Malang, Indonesia',
  style: TextStyle(...),
),
],
),
),
/* soal 3 */
Icon(
  ...,
  color: ...,
),
const Text(...),
],
),
);
```

/* soal 1 */ Letakkan widget `Column` di dalam widget `Expanded` agar menyesuaikan ruang yang tersisa di dalam widget `Row`. Tambahkan properti `crossAxisAlignment` ke `CrossAxisAlignment.start` sehingga posisi kolom berada di awal baris.

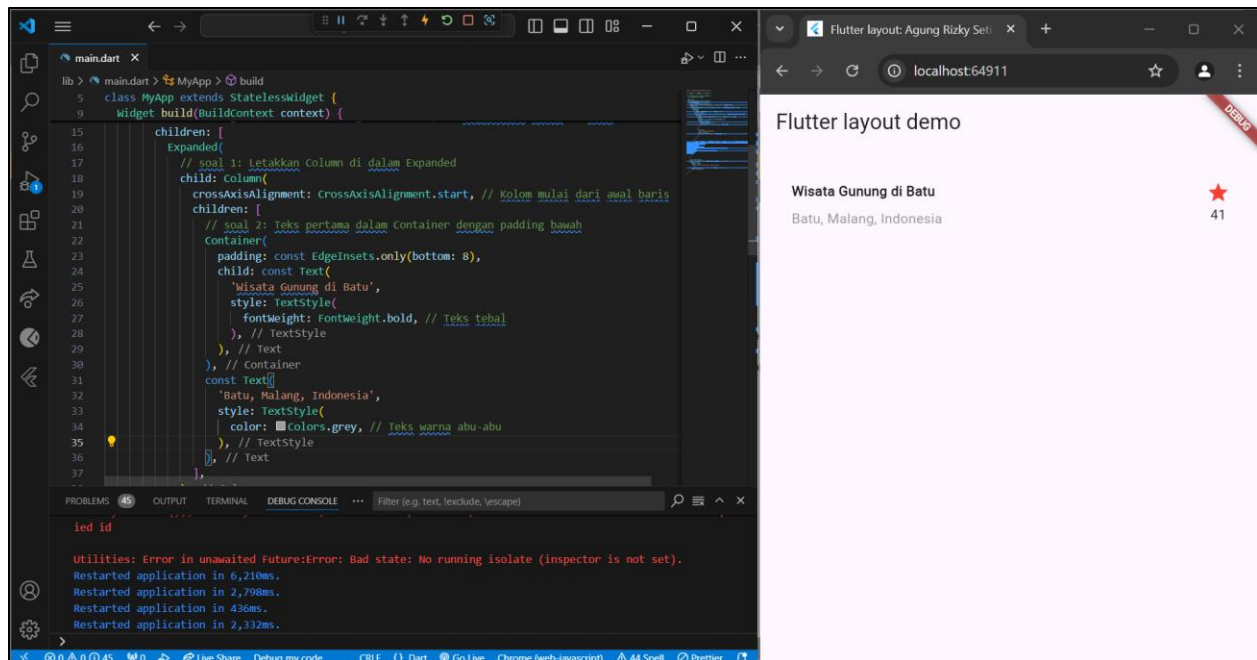
/* soal 2 */ Letakkan baris pertama teks di dalam `Container` sehingga memungkinkan Anda untuk menambahkan padding = 8. Teks 'Batu, Malang, Indonesia' di dalam `Column`, set warna menjadi abu-abu.

/* soal 3 */ Dua item terakhir di baris judul adalah ikon bintang, set dengan warna merah, dan teks "41". Seluruh baris ada di dalam `Container` dan beri padding di sepanjang setiap tepinya sebesar 32 piksel. Kemudian ganti isi `body` text 'Hello World' dengan variabel `titleSection` seperti berikut:



```
@@ -14,11 +48,13 @@
14 48      return MaterialApp(
15 49        title: 'Flutter layout demo',
16 50        home: Scaffold(
17 51          appBar: AppBar(
18 52            title: const Text('Flutter layout demo'),
19 53          ),
20 -      body: const Center(
21 -        child: Text('Hello World'),
22 +      body: Column(
23 +        children: [
24 +          titleSection,
25 +          Icon(Icons.star, color: Colors.red),
26 +          Text('41'),
27 +        ],
28      ),
29    ),
30  );
```

Jawaban :



Praktikum 2: Implementasi button row

Langkah 1: Buat method `Column _buildButtonColumn`

Bagian tombol berisi 3 kolom yang menggunakan tata letak yang sama—sebuah ikon di atas baris teks. Kolom pada baris ini diberi jarak yang sama, dan teks serta ikon diberi warna primer.

Karena kode untuk membangun setiap kolom hampir sama, buatlah metode pembantu pribadi bernama `buildButtonColumn()`, yang mempunyai parameter warna, Icon dan Text, sehingga dapat mengembalikan kolom dengan widgetnya sesuai dengan warna tertentu.

lib/main.dart (`_buildButtonColumn`)

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

  @override
  Widget build(BuildContext context) {
    // ...
  }

  Column _buildButtonColumn(Color color, IconData icon, String label) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.min,
      mainAxisAlignment: MainAxisAlignment.center,
```

```

children: [
  Icon(icon, color: color),
  Container(
    margin: const EdgeInsets.only(top: 8),
    child: Text(
      label,
      style: TextStyle(
        fontSize: 12,
        fontWeight: FontWeight.w400,
        color: color,
      ),
    ),
  ),
],
);
}
}

```

Langkah 2: Buat widget buttonSection

Buat Fungsi untuk menambahkan ikon langsung ke kolom. Teks berada di dalam `Container` dengan margin hanya di bagian atas, yang memisahkan teks dari ikon.

Bangun baris yang berisi kolom-kolom ini dengan memanggil fungsi dan set warna, `Icon`, dan teks khusus melalui parameter ke kolom tersebut. Sejajarkan kolom di sepanjang sumbu utama menggunakan `MainAxisAlignment.spaceEvenly` untuk mengatur ruang kosong secara merata sebelum, di antara, dan setelah setiap kolom. Tambahkan kode berikut tepat di bawah deklarasi `titleSection` di dalam metode `build()`:

lib/main.dart (buttonSection)

```

Color color = Theme.of(context).primaryColor;

Widget buttonSection = Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    _buildButtonColumn(color, Icons.call, 'CALL'),
    _buildButtonColumn(color, Icons.near_me, 'ROUTE'),
    _buildButtonColumn(color, Icons.share, 'SHARE'),
  ],
);

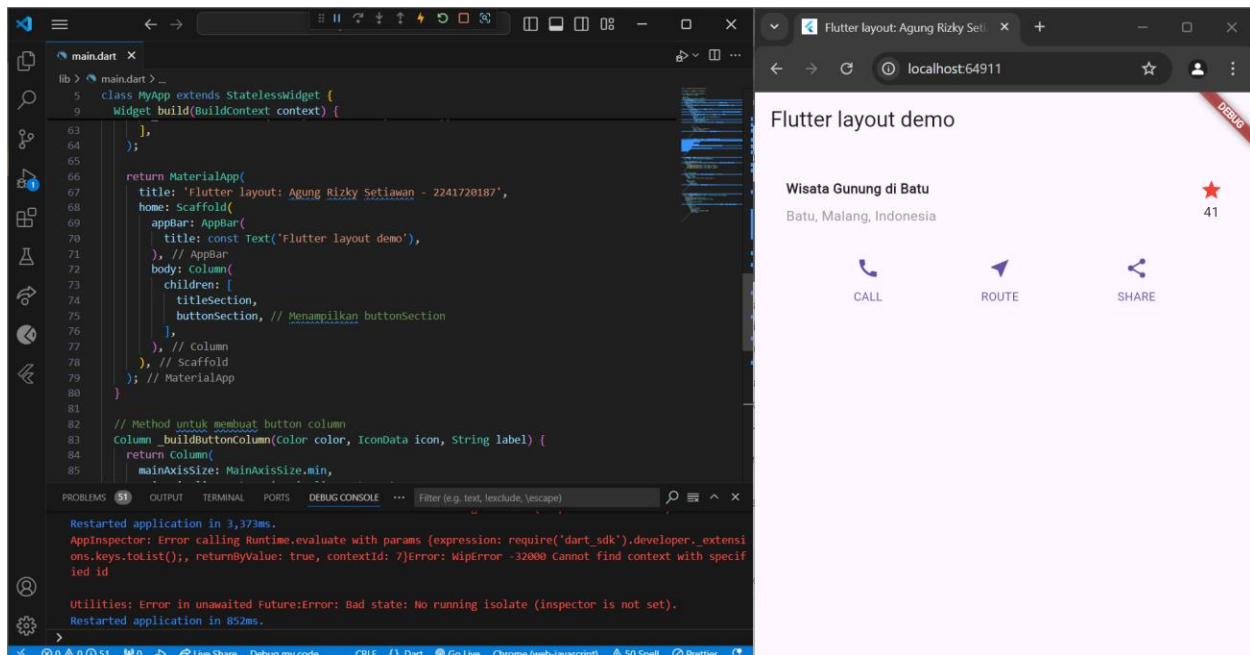
```

Langkah 3: Tambah button section ke body

Tambahkan variabel `buttonSection` ke dalam `body` seperti berikut:

{step2 → step3}/lib/main.dart		
Viewed		
		@@ -48,3 +59,3 @@
48	59	return MaterialApp(
49	60	title: 'Flutter layout demo',
50	61	home: Scaffold(
		@@ -54,8 +65,9 @@
54	65	body: Column(
55	66	children: [
56	67	titleSection,
	68	+ buttonSection,
57	69],
58	70),
59	71),
60	72);
61	73	}

Jawaban :



Praktikum 3: Implementasi text section

Langkah 1: Buat widget textSection

Tentukan bagian teks sebagai variabel. Masukkan teks ke dalam `Container` dan tambahkan padding di sepanjang setiap tepinya. Tambahkan kode berikut tepat di bawah deklarasi `buttonSection`:

```
Widget textSection = Container(  
  padding: const EdgeInsets.all(32),  
  child: const Text(  
    'Carilah teks di internet yang sesuai '  
    'dengan foto atau tempat wisata yang ingin '  
    'Anda tampilkan. '  
    'Tambahkan nama dan NIM Anda sebagai '  
    'identitas hasil pekerjaan Anda. '  
    'Selamat mengerjakan 😊.',  
    softWrap: true,  
  ),  
);
```

Dengan memberi nilai `softWrap` = true, baris teks akan memenuhi lebar kolom sebelum membungkusnya pada batas kata.

Langkah 2: Tambahkan variabel text section ke body

Tambahkan widget variabel `textSection` ke dalam `body` seperti berikut:

{step3 → step4}/lib/main.dart		
<input type="checkbox"/> Viewed		
@@ -59,3 +72,3 @@		
59	72	return MaterialApp(60 73 title: 'Flutter layout demo', 61 74 home: Scaffold(@@ -66,6 +79,7 @@ 66 79 children: [67 80 titleSection, 68 81 buttonSection, 82 + textSection, 69 83], 70 84), 71 85),

Jawaban :

- File pubspec juga sensitif terhadap spasi, jadi gunakan indentasi yang tepat.
- Anda mungkin perlu memulai ulang program yang sedang berjalan (baik di simulator atau perangkat yang terhubung) agar perubahan pubspec dapat diterapkan.

Langkah 2: Tambahkan gambar ke body

Tambahkan aset gambar ke dalam `body` seperti berikut:

		{step4 → step5}/lib/main.dart	
		<input type="checkbox"/> Viewed	
		@@ -77,6 +77,12 @@	
77	77),	
78	78	body: Column(
79	79	children: [
80	80	+	Image.asset(
81	81	+	'images/lake.jpg',
82	82	+	width: 600,
83	83	+	height: 240,
84	84	+	fit: BoxFit.cover,
85	85	+),
80	86	titleSection,	
81	87	buttonSection,	
82	88	textSection,	

`BoxFit.cover` memberi tahu kerangka kerja bahwa gambar harus sekecil mungkin tetapi menutupi seluruh kotak rendernya.

Langkah 3: Terakhir, ubah menjadi ListView

Pada langkah terakhir ini, atur semua elemen dalam `ListView`, bukan `Column`, karena `ListView` mendukung scroll yang dinamis saat aplikasi dijalankan pada perangkat yang resolusinya lebih kecil.

{step5 → step6}/lib/main.dart		
Viewed		
@@ -72,13 +77,13 @@		
72	77	return MaterialApp(
73	78	title: 'Flutter layout demo',
74	79	home: Scaffold(
75	80	appBar: AppBar(
76	81	title: const Text('Flutter layout demo'),
77	82),
78	-	body: Column(
83	+	body: ListView(
79	84	children: [
80	85	Image.asset(
81	86	'images/lake.jpg',
82	87	width: 600,
83	88	height: 240,
84	89	fit: BoxFit.cover,

Jawaban :

