Nama : E A Yoga Wilanda

NIM : 1201222103

Tugas 3: Layout Part 1

Widget

- **Penjelasan:** Widget adalah komponen dasar dari Flutter, Widget mewakilkan apa yang akan dilihat atau akan digunakan. Ada 2 tipe Widget, yaitu Stateless dan StatefulWidget.

Container

- **Penjelasan:** Nested Row berarti meletakkan **Row** di dalam **Row** lain atau menggabungkan **Row** dengan **Column** untuk tata letak yang lebih kompleks. Ini memungkinkan Anda membuat struktur layout yang fleksibel.

- **Contoh Kode :**

```
Container(
  height: 100, // pengaturan ukuran ketinggian
  width: 100, // pengaturan ukuran panjang (menyamping)
  color: Colors.blue, //pengaturan untuk membuat warna
  child: Center(child: Text("Container")),
);
```

Row & Column

- **Penjelasan:** Secara default, Row akan digunakan untuk membuat tata letak sejumlah kumpulan widget menjadi tertata horizontal/menyamping, dan Column akan menjadi

vertikal/menurun. Kelebihan Row & Column adalah dapat menampul kumpulan widget karena turunan widgetnya menggunakan children, yang dimana dapat berisi lebih dari 1.

- **Contoh Kode :** Membuat dua kolom di dalam satu baris. Tiap kolom memiliki dua baris teks.

ListView

- **Penjelasan:** Listview membuat widget yang ada di dalamnya itu bisa digulir/scroll, untuk pengaturan scroll dapat dikonfigurasikan menjadi vertikal atau horizontal.

  Listview dapat mengakibatkan overflow dari widget yang memiliki fix size, jadi, solusinya bungkus dengan Expanded, atau Flexible Widget.

  Contoh sederhananya saya menggunakan ListView.builder, supaya auto generate Listnya.

- **Contoh Kode :**

```
ListView.builder(
  itemCount: 10, // membuat listnya hanya berisi 10 saja
//item builder wajib, karena isi dari listnya adalah adalah widget apa yang akan dijadikan list, contoh, card, ListTile, Container, dll.
  itemBuilder: (context, index) {
    return ListTile(
      title: Text("Item $index"),
    );
  },
);
```

GridView

- **Penjelasan:** GridView adalah layout yang menampilkan widget dalam format grid (baris dan kolom). Hal ini sangat berguna untuk galeri atau daftar dengan tata letak kotak. Kurang lebih sama dengan ListView hanya saja modelnya seperti Grid/Tampilan Galeri.

- **Contoh Kode :**

```
GridView.count(
  crossAxisCount: 2, // membuat aturan 2 kotak maximal secara vertikal
  children: [
    Container(color: Colors.red),
    Container(color: Colors.blue),
    Container(color: Colors.green),
    Container(color: Colors.yellow),
  ],
);
```

Stack

- **Penjelasan:** Stack memungkinkan turunan/anak dari widgetnya dapat ditumpuk dalam 1 koordinat yang sama, sehingga secara visual bisa terlihat tumpang tindih.

- **Contoh Kode :**

```
Stack(
  alignment: Alignment.center,
  children: [
```

```
    Container(width: 200, height: 200, color: Colors.red),
    Container(width: 150, height: 150, color: Colors.green),
    Container(width: 100, height: 100, color: Colors.blue),
  ],
);.
```

Tugas 4: Layout Part 2 (Lanjutan)

Nested Row

- **Penjelasan:** Nested Row berarti meletakkan **Row** di dalam **Row** lain atau menggabungkan **Row** dengan **Column** untuk tata letak yang lebih kompleks. Ini memungkinkan Anda membuat struktur layout yang fleksibel.

- **Kode Contoh:** Membuat dua kolom di dalam satu baris. Tiap kolom memiliki dua baris teks.

```
Row(
  children: [
    Column(
      children: [
        Text("Column 1, Row 1"),
        Text("Column 1, Row 2"),
      ],
    ),
    Column(
      children: [
        Text("Column 2, Row 1"),
        Text("Column 2, Row 2"),
```

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

TabView

- **Penjelasan: TabView** digunakan untuk membuat navigasi berbasis tab, seperti pada aplikasi dengan banyak kategori. Biasanya digabungkan dengan **DefaultTabController**.

- **Kode Contoh:** Membuat 3 tab dengan konten berbeda. **TabBar** di bagian atas, dan **TabBarView** untuk kontennya.

```
DefaultTabController(
  length: 3,
  child: Scaffold(
    appBar: AppBar(
      bottom: TabBar(
        tabs: [
          Tab(text: "Tab 1"),
          Tab(text: "Tab 2"),
          Tab(text: "Tab 3"),
        ],
      ),
      title: Text("TabView Example"),
    ),
    body: TabBarView(
```

```
      children: [
        Center(child: Text("Content 1")),
        Center(child: Text("Content 2")),
        Center(child: Text("Content 3")),
      ],
    ),
  ),
);
```

PageView

- **Penjelasan: PageView** digunakan untuk membuat tampilan halaman yang dapat digeser (swipe) secara horizontal atau vertikal. Biasanya untuk carousel atau wizard.

- **Kode Contoh:** Membuat 3 halaman dengan warna berbeda.

```
PageView(
  children: [
    Container(color: Colors.red, child: Center(child: Text("Page 1"))),
    Container(color: Colors.blue, child: Center(child: Text("Page 2"))),
    Container(color: Colors.green, child: Center(child: Text("Page 3"))),
  ],
);
```

SafeArea

- **Penjelasan: SafeArea** memastikan bahwa widget tidak terpotong oleh notch, status bar, atau elemen UI lainnya. Berguna untuk membuat layout yang aman di berbagai perangkat.

- **Kode Contoh:** Membungkus konten dengan **SafeArea** untuk memastikan tidak tertabrak status bar.

```
SafeArea(
  child: Scaffold(
    appBar: AppBar(title: Text("Safe Area Example")),
    body: Center(child: Text("Content is safe from notches!")),
  ),
);
```

Tugas 5: User Interaction

Stateful & Stateless Widget

- **Penjelasan:**
  - **Stateless Widget**: Tidak memiliki keadaan (state) yang berubah. Cocok untuk UI yang statis, seperti teks atau gambar.
  - **Stateful Widget**: Memiliki state yang dapat diubah, sehingga cocok untuk UI yang dinamis, seperti tombol yang memperbarui nilai.

- **Kode Contoh:** Contoh implementasi **Stateless** untuk teks, dan **Stateful** untuk tombol yang mengubah nilai counter.

```
// Stateless Widget
class MyStatelessWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
    return Text("I am Stateless");
  }
}
```

// Stateful Widget ← yang hanya bisa mengakses function setState secara native untuk perubahan state di dalam widget.

```
class MyStatefulWidget extends StatefulWidget {
  @override
  _MyStatefulWidgetState createState() => _MyStatefulWidgetState();
}

class _MyStatefulWidgetState extends State<MyStatefulWidget> {
  int counter = 0;

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Text("Counter: $counter"), // $counter akan diperhatikan untuk perubahan datanya.
        ElevatedButton(
          onPressed: () {
            setState(() { // function setState ini yang bisa mengubah state dari Statefull
              Counter++; // pengubah state
            });
          },
```

```
          child: Text("Increment"),
        ),
      ],
    );
  }
}
```

Button

- **Penjelasan:** Tombol adalah widget yang dapat diklik untuk melakukan tindakan tertentu. Flutter menyediakan berbagai jenis tombol seperti **ElevatedButton**, **TextButton**, dan **OutlinedButton**.

- **Kode Contoh:** Membuat **ElevatedButton** dengan aksi sederhana mencetak teks ke konsol.

```
ElevatedButton(
  onPressed: () {
    print("Button Pressed!"); // tidak direkomendasikan menggunakan print di produksi, gunakan debugPrint jika ingin lebih aman.
  },
  child: Text("Click Me"),
);
```

Snackbar

- **Penjelasan: Snackbar** digunakan untuk menampilkan pesan sementara di bagian bawah layar. Biasanya digunakan untuk konfirmasi atau pemberitahuan singkat.

- **Kode Contoh:** Menampilkan snackbar dengan pesan "Hello Snackbar!".

```
ScaffoldMessenger.of(context).showSnackBar(
  SnackBar(content: Text("Hello Snackbar!")),
);
```

Dialog

- **Penjelasan: Dialog** digunakan untuk menampilkan pop-up seperti konfirmasi atau formulir singkat. Biasanya berbentuk **AlertDialog**.

- **Kode Contoh:** Menampilkan dialog dengan judul, konten, dan tombol aksi untuk menutup dialog.

```
showDialog(
  context: context,
  builder: (BuildContext context) {
    return AlertDialog(
      title: Text("Dialog Title"),
      content: Text("This is a dialog."),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: Text("OK"),
        ),
```

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

Menu (Titik 3)

- **Penjelasan: PopupMenuButton** adalah menu konteks (biasanya berupa 3 titik vertikal) yang menampilkan daftar opsi saat diklik.

- **Kode Contoh:** Membuat menu dengan 2 opsi yang dapat dipilih.

```
PopupMenuButton<String>(
  onSelected: (value) => print("Selected: $value"),
  itemBuilder: (context) => [
    PopupMenuItem(value: "Option 1", child: Text("Option 1")),
    PopupMenuItem(value: "Option 2", child: Text("Option 2")),
  ],
);
```

Tugas 6: Navigasi & Notifikasi

Package

- **Penjelasan:** Package dalam Flutter adalah library eksternal yang bisa menambahkan fitur tambahan ke aplikasi. Contohnya, menggunakan package http untuk mengambil data dari API.

- **Kode Contoh:** Menggunakan package http untuk mengambil data dari URL.

  Cara menginstall package cari nama/kode nama dependencies dari library yang dibutuhkan

  Lalu sebagai contoh http, untuk install ketik di command prompt/terminal

  flutter pub add http

  ```
  // contoh menggunakan http
  // as digunakan untuk membuat nama samaran/alias
  import 'package:http/http.dart' as http;

  Future<void> fetchData() async {
    var response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/posts/1'));
    print(response.body);
  }
  ```

Navigation

- **Penjelasan:** Navigasi digunakan untuk berpindah antar halaman (screen) dalam aplikasi. Flutter menggunakan **Navigator** untuk menangani perpindahan ini.

- **Kode Contoh:** Menggunakan **Navigator.push** untuk membuka halaman baru.

  ```
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => SecondScreen()),
  );
  ```

Notification

- **Penjelasan:** Pemberitahuan memungkinkan aplikasi mengirim pesan ke pengguna bahkan ketika aplikasi sedang tidak aktif. Package seperti flutter_local_notifications memungkinkan pengaturan notifikasi lokal.
- **Kode Contoh:** Menampilkan notifikasi sederhana menggunakan package flutter_local_notifications.

```
// contoh menggunakan local notification dari flutter_local_notifications

import 'package:flutter_local_notifications/flutter_local_notifications.dart';



FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =
FlutterLocalNotificationsPlugin();



void showNotification() {

  var android = AndroidNotificationDetails('id', 'name', 'description');

  var platform = NotificationDetails(android: android);

  flutterLocalNotificationsPlugin.show(0, 'Hello', 'This is a notification', platform);

}
```