Vivian Wijaya (0806022410002)

Flutter Provider Report

1. Introduction

Assignment ini dikerjakan dengan menggunakan flutter untuk membuat aplikasi counter yang interaktif dan fleksibel. App ini memungkinkan user untuk menambahkan lebih banyak counter sekaligus sesuai yang diinginkan, juga memungkinkan user untuk melakukan berbagai aksi seperti increment, decrement, mengubah warna, dan mengubah/mengedit label nama di setiap counternya.

2. Part 1: Local State Management

Di dalam tahap pertama ini, saya membuat counter yang sederhana dengan menggunakan StatefulWidget untuk counternya. Semua data dari counter tersebut akan disimpan langsung di dalam widget itu sendiri. Pada tahap ini, counter bisa ditambah (increment) atau dikurangi (decrement) nilainya, tapi nilainya tidak bisa kurang dari nol, sehingga tidak akan muncul angka negatif saat digunakan.

```dart
class _CounterWidgetState extends State<CounterWidget> {
  int _counter = 0;

  void _increment() {
    setState(() {
      _counter++;
    });
  }

  void _decrement() {
    setState(() {
      if (_counter > 0) {
        _counter--;
      }
    });
  }
```

3. Part 2: Global State Management

Setelah itu, saya membuat global state menggunakan provider supaya data counter bisa dipakai di banyak widget sekaligus. Di sini, saya membuat CounterProvider yang menyimpan semua daftar counter. Dengan provider ini, saya bisa melakukan CRUD menambah, menghapus, menaikkan, atau menurunkan nilai counter tersebut secara mudah. Semua counter yang ada juga

ditampilkan di layar menggunakan ReorderableListView, jadi pengguna bisa mengatur urutan counter dengan drag & drop sesuai keinginan.

```
// Tombol tambah counter
floatingActionButton: FloatingActionButton(
  child: const Icon(Icons.add),
  onPressed: model.addCounter,
), // FloatingActionButton
```

Tombol ini memanggil addCounter() di CounterProvider untuk menambahkan counter baru ke daftar. Begitu ditekan, daftar counter otomatis terupdate karena notifyListeners() dipanggil di provider.

```
ReorderableListView(
  onReorder: model.reorder,
  children: [
    for (int i = 0; i < model.counters.length; i++)
      CounterTile(
        key: ValueKey(
          model.counters[i],
        ), // Key unik agar reorder aman // ValueKey
        index: i,
      ), // CounterTile
  ],
), // ReorderableListView
```

CounterTile adalah widget untuk tiap counter (menampilkan label, value, tombol aksi). ValueKey(model.counters[i]) merupakan key unik untuk setiap tile agar ReorderableListView bisa bekerja tanpa error.

4. Part 3: Advanced UI and Interactivity

Di bagian ini, saya menambahkan beberapa fitur supaya UI lebih menarik dan interaktif. Setiap counter punya warna yang berbeda-beda supaya gampang dibedakan. Pengguna juga bisa mengubah nama counter atau mengganti warnanya lewat dialog yang muncul saat tombol ditekan. Selain itu, saat menekan tombol increment atau decrement, perubahan nilai counter tampil dengan animasi halus menggunakan AnimatedSwitcher dan ScaleTransition jadi terlihat lebih menarik. Tidak hanya itu, user juga bisa mengatur urutan counter dengan drag & drop, sehingga tampilan list dari counter tersebut bisa disesuaikan sesuai kebutuhan.

```
AnimatedSwitcher(
  duration: const Duration(milliseconds: 300),
  transitionBuilder: (child, anim) =>
      ScaleTransition(scale: anim, child: child),
```

AnimatedSwitcher digunak untuk membuat animasi menjadi lebih smooth ketika counternya berubah.

```
IconButton(
  icon: const Icon(Icons.color_lens, color: Colors.purple),
  onPressed: () {
    Color pickerColor = counter.color;
```

IconButton dan ColorPicker diatas digunakan untuk interaksi antara app dengan pengguna, seperti contohnya jika si pengguna ingin mengubah warna

5. Challenges

- **ReorderableListView harus pakai Key unik**

  Awalnya error terus saat drag & drop karena widget tidak punya Key. Setelah tiap CounterTile pakai ValueKey dari index, Flutter bisa mengenali setiap tile secara unik, sehingga drag & drop berjalan lancar.

- **Counter tidak boleh minus**

  Awalnya tombol decrement bisa ditekan terus walau counter = 0, tapi kita ingin nilai counter tetap aman. Solusinya: sebelum decrement, kita cek dulu if counter > 0 Jadi:

  - Kalau counter = 0 → tombol ditekan → nilai tetap 0
  - Kalau counter > 0 → tombol ditekan → nilai berkurang 1

Dengan cara ini, counter tidak pernah negatif, tapi interaksi tetap smooth.

```
void _decrement() {
  setState(() {
    if (_counter > 0) {
      _counter--;
    }
  });
}
```

6. Advanced Features Implemented

- **Kelola Counter (CRUD)**

Jadi user bisa menambahkan counter baru sebanyak yang diinginkan bisa increment dan decrement, juga bisa mengubah nama counter sesuai yang diinginkan dan mengganti warna. Dimana hal tersebut bisa dilakukan dengan mudah oleh user melalui tombol-tombol yang telah tersedia di app tersebut

- **Reorderable List**

  Counter tersebut bisa diubah susunannya oleh user