Refleksi Flutter Provider

Visual Programming

Nama : Michele Stevany Venda Dati

NIM : 0806022410021

A. Pendahuluan

Project ini bertujuan untuk membangun aplikasi counter di Flutter yang menunjukkan kemampuan *advanced state management,* baik secara lokal maupun global (2 aplikasi yang berbeda). Proyek dibagi menjadi empat tahap utama, antara lain:

- Step 1: Berfokus pada *local state management.*
- Step 2: Berfokus pada *global state management* menggunakan provider.
- Step 3: Pengembangan UI interaktif dengan animasi, edit label, ganti warna dan *reorderable list.*
- Step 4: Dokumentasi dan refleksi.

B. Local State - Step 1

Di tahap awal, saya membuat counter sederhana menggunakan `StatefulWidget` (lanjutan lab activity week 4). Fitur yang diterapkan berupa tombol increment dan decrement dengan logika nilai counter tidak bisa kurang dari nol. *Local state* hanya efektif untuk widget tunggal. Ketika state ingin diakses oleh banyak widget, pendekatan ini menjadi terbatas dan tidak *scalable.*

- `StatefulWidget` digunakan karena state hanya relevan untuk satu widget.
- `setState()` dipanggil setiap kali nilai counter berubah agar widget di*rebuild.*
- Counter disimpan sebagai variabel interger dalam `_CounterWidgetState`.

C. Global State - Step 2

Tahap ini memperkenalkan *global state management* dengan membuat class `GlobalState` yang extends `ChangeNotifier`. State ini disebarkan ke seluruh widget menggunakan package `Provider`. Fitur yang implementasikan adalah `addCounter`

(menambahkan counter baru), `removeCounter` (menghapus counter), `increment` dan `decrement` (increment dan decrement tiap counter).

- `GlobalState` menyimpan daftar counter sebagai `List<CounterModel>`.
- Setiap method memanggil `notifyListeners()` untuk memberitahu Provider agar memperbarui UI.
- `ChangeNotifierProvider` *wrap* root widget di `main.dart` agar state global bisa diakses semua widget.
- `Uuid` digunakan untuk menghasilkan ID unik tiap counter agar dapat dikelola secara individual.

D. Global State - Step 3

Di bagian ini, saya menambahkan animasi dan beberapa fitur interaktif untuk meningkatkan *user experience*. Ada beberapa fitur yang dimplementasikan, yaitu `AnimatedContainer` (transisi warna *smooth*), `AnimatedSwitcher` (animasi perubahan nilai counter), `AlertDialog` dengan `TextField` (mengubah label counter), `PopupMenuButton<Color>` (*popup menu* untuk memilih warna counter), dan `ReorderableListView` (mengubah urutan counter dengan *drag-and-drop*).

E. Tantangan dan Pembelajaran

Terdapat beberapa tantangan yang saya hadapi. Awalnya, struktur folder dan package untuk `GlobalState` sempat menjadi masalah karena saya mencoba untuk membuat package terpisah yang *reusable* (`flutter create --template=package global_state`), sehingga *path* dan *dependency* harus diatur dengan hati-hati dan teliti agar dapat diakses oleh aplikasi utama.

Selain itu, saya mendapat error pada `ChangeNotifierProvider` (`The argument type 'GlobalState Function(BuildContext)' can't be assigned to the parameter type 'Create<ChangeNotifier?>'`) karena menulis `create: GlobalState()`. Masalahnya, `create` harus menerima *function* yang mengembalikan instance, bukan instance langsung. Solusinya, saya menulis `create: () => GlobalState()` dan

menambahkan tipe generic `<GlobalState>`. Dengan begitu, provider dapat mengenali tipe state, dan semua widget di dalamnya bisa mengakses state global dengan benar.

Kemudian, saat menambahkan animasi pada increment dan decrement, terdapat *glitch* dimana awalnya, jika `Text` yang menampilkan nilai counter tidak memiliki *unique key,* sehingga `AnimatedSwitcher` kesulitan untuk membedakan widget lama dan baru. Untuk mengatasinya, saya menggunakan `ValueKey(counter.value)` pada widget `Text`, sehingga setiap perubahan nilai counter dianggap widget baru oleh `AnimatedSwitcher`.

F. Kesimpulan

Project ini menunjuukan perkembangan *state management* di Flutter dari *local* ke global. *Local State* cocok untuk widget tunggal, sedangkan *global state* dengan Provider memudahkan pengelolaan data di seluruh aplikasi. Penambahan animasi, edit label, pemilihan warna, dan *reorderable list* meningkatkan interaktivitas dan *user experience.* Tantangan seperti sinkronisasi state dengan UI dan penggunaan *unique key* memberi pembelajaran penting mengenai modularisasi, *type-safety,* dan desain aplikasi Flutter yang responsif dan lebih *scalable.*