

**PRAKTIKUM PEMROGRAMAN PERANGKAT BERGERAK
TUGAS GUIDED & UNGUIDED**

**MODUL XIII
NETWORKING**



Disusun Oleh :

Rizky Hanifa Afania / 2211104017

SE-06-01

Asisten Praktikum :

Muhammad Faza Zulian Gesit Al Barru

Aisyah Hasna Aulia

Dosen Pengampu :

Yudha Islami Sulistya, S.Kom., M.Cs.

**PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

GUIDED

NETWORKING/STATE MANAGEMENT

Manajemen state dalam Flutter adalah proses pengelolaan status atau kondisi aplikasi, yaitu data atau informasi yang dapat berubah selama siklus hidup aplikasi. State ini mencakup berbagai aspek yang memengaruhi tampilan antarmuka pengguna (UI), seperti input pengguna, data dari API, dan status internal widget. Seiring meningkatnya kompleksitas aplikasi, akan tiba saat di mana state perlu dibagikan antar halaman.

Flutter menggunakan pendekatan deklaratif, di mana UI dibangun berdasarkan state yang sedang berlaku. Dengan menerapkan manajemen state, semua state dari berbagai kontrol UI dapat disentralisasi untuk mengelola aliran data di seluruh aplikasi.

Manajemen state memiliki peran penting karena aplikasi Flutter biasanya terdiri dari banyak widget yang saling berhubungan. Pengelolaan state yang baik memungkinkan:

- Sinkronisasi UI dengan data, memastikan UI selalu mencerminkan data terkini
- Organisasi kode yang terstruktur, sehingga memudahkan pengembangan dan pemeliharaan.
- Pengurangan bug, karena state yang terkelola dengan baik dapat meminimalkan potensi kesalahan.

Jenis State dalam Flutter

1. Ephemeral State (State Lokal)

State ini hanya berlaku untuk widget tertentu dan tidak perlu dibagikan ke widget lainnya. Contohnya adalah state yang digunakan pada widget seperti TextField atau Checkbox. Untuk mengelola jenis state ini, kita dapat memanfaatkan StatefulWidget, yang cocok untuk menangani ephemeral state.

Terdapat dua pendekatan utama dalam manajemen state:

- StatefulWidget untuk mengelola ephemeral state yang bersifat lokal pada widget tertentu.
- InheritedWidget untuk berbagi state di antara beberapa widget.

2. App State (State Global)

State ini digunakan oleh berbagai widget di dalam aplikasi. Contohnya meliputi informasi pengguna yang sedang masuk, data keranjang belanja, atau pengaturan tema aplikasi. App state umumnya memerlukan pendekatan manajemen state yang lebih kompleks. Untuk mendukung hal ini, Flutter menyediakan berbagai framework atau pustaka untuk manajemen state, seperti:

A. Provider

Provider adalah library state management yang didukung resmi oleh tim Flutter. Provider memanfaatkan kemampuan bawaan Flutter seperti InheritedWidget, tetapi dengan cara yang lebih sederhana dan efisien

B. BloC/Cubit

Bloc (Business Logic Component) adalah pendekatan state management berbasis pola stream. Bloc memisahkan business logic dari UI, sehingga cocok untuk aplikasi yang besar dan kompleks.

C. Riverpod

Riverpod adalah framework state management modern yang dirancang sebagai pengganti atau alternatif untuk Provider. Riverpod lebih fleksibel dan mengatasi beberapa keterbatasan Provider.

D. GetX

GetX adalah framework Flutter serbaguna yang menyediakan solusi lengkap untuk state management, routing, dan dependency injection. GetX dirancang untuk meminimalkan boilerplate code, meningkatkan efisiensi, dan mempermudah pengembangan aplikasi Flutter, terutama yang memerlukan reaktivitas tinggi

Instalasi GetX

1. Tambahkan GetX ke dalam proyek Flutter melalui pubspec.yaml

```
dependencies:  
  flutter:  
    sdk: flutter  
  cupertino_icons: ^1.0.8  
  get: ^4.6.6
```

2. Konfigurasi dasar

```
import 'package:flutter/material.dart';  
import 'package:get/get.dart';  
import 'package:modul13/view/detail_page.dart';  
import 'package:modul13/view/my_home_page.dart';  
  
void main() {  
  runApp(const MyApp());  
}  
  
class MyApp extends StatelessWidget {
```

```

const MyApp({super.key});

@override
Widget build(BuildContext context) {
  return GetMaterialApp(
    title: 'Praktikum 13 - GetX',
    home: MyHomePage(),
  );
}

```

3. State Management dengan GetX

a. Membuat Controller

- Tambahkan controller ke dalam widget menggunakan Get.put() untuk dependency injection.
- Gunakan Obx untuk memantau perubahan state.

```

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

class CounterController extends GetxController {
  var counter = 0.obs;

  //Fungsi untuk menambah
  void incrementCounter() {
    counter++;
  }
}

```

b. Menggunakan Controller di UI

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul13/view_model/controller.dart';

class MyHomePage extends StatelessWidget {
  MyHomePage({super.key, required this.title});

  final String title;
  final CounterController controller = Get.put(CounterController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(

```

```

appBar: AppBar(
  backgroundColor: Theme.of(context).colorScheme.inversePrimary,
  title: Text(title),
),
body: Center(
  child: Obx(
    () => Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        const Text(
          'You have pushed the button this many times:',
        ),
        Text(
          controller.counter.toString(),
          style: Theme.of(context).textTheme.headlineMedium,
        ),
        ElevatedButton(
          onPressed: () {
            Get.toNamed('/detail');
          },
          child: Text('Ke Halaman Detail'))
      ],
    ),
  ),
),
floatingActionButton: Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    FloatingActionButton(
      onPressed: controller.incrementCounter,
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ),
  ],
),
);
}
}

```

4. Routing dengan GetX

a. Definisikan Route

Gunakan GetPage pada main.dart untuk mendefinisikan rute aplikasi :

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul13/view/detail_page.dart';
import 'package:modul13/view/my_home_page.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      initialRoute: '/',
      getPages: [
        GetPage(name: '/', page: () => MyHomePage(title: 'Belajar
GetX')),
        GetPage(name: '/detail', page: () => DetailPage()),
      ],
    );
  }
}

```

b. Navigasi

- `Get.to()` : Navigasi ke halaman baru.
- `Get.back()` : Kembali ke halaman sebelumnya.
- `Get.off()` : Menghapus semua halaman sebelumnya.
- `Get.offAll()` : Menghapus semua halaman dalam stack.

5. Dependency Injection dengan GetX

a. Injeksi Sederhana

Gunakan `Get.put()` untuk membuat instance controller yang tersedia di mana saja :

```
final CounterController controller = Get.put(CounterController());
```

b. Lazy Loading

Gunakan `Get.lazyPut()` jika ingin membuat instance hanya saat dibutuhkan :

```
final CounterController controller =
Get.lazyPut(CounterController());
```

c. Mengambil Instance

Ambil instance di mana saja dalam aplikasi :

```
final CounterController controller = Get.find(CounterController());
```

6. SnackBar

```
void getSnackBar() {  
  Get.snackbar(  
    'GetX Snackbr',  
    'Ini adalah getX',  
    backgroundColor: Colors.amber,  
    colorText: Colors.black,  
  );  
}
```

7. Dialog

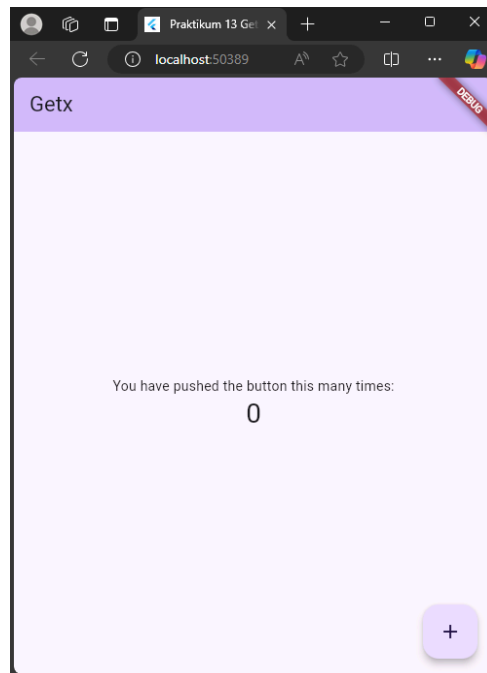
```
void getDialog() {  
  Get.defaultDialog(  
    title: 'Dialog Title',  
    middleText: 'This is a dialog',  
  );  
}
```

8. BottomSheet

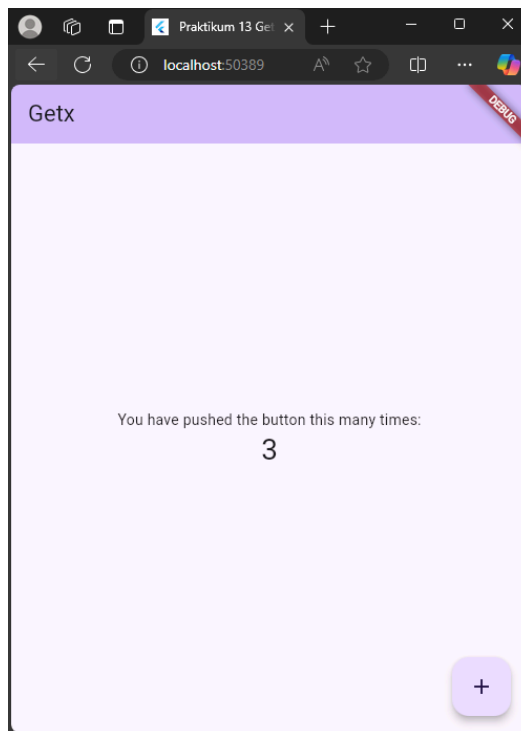
```
void getbottomshet() {  
  Get.bottomSheet(  
    Container(  
      height: 200,  
      color: Colors.amber,  
      child: const Center(  
        child: Text(  
          'Ini adalah getX BottomSheet',  
          style: TextStyle(  
            color: Colors.black,  
            fontSize: 20,  
          ),  
        ),  
      ),  
    ),  
  );  
}
```

Hasil Output Praktikum:

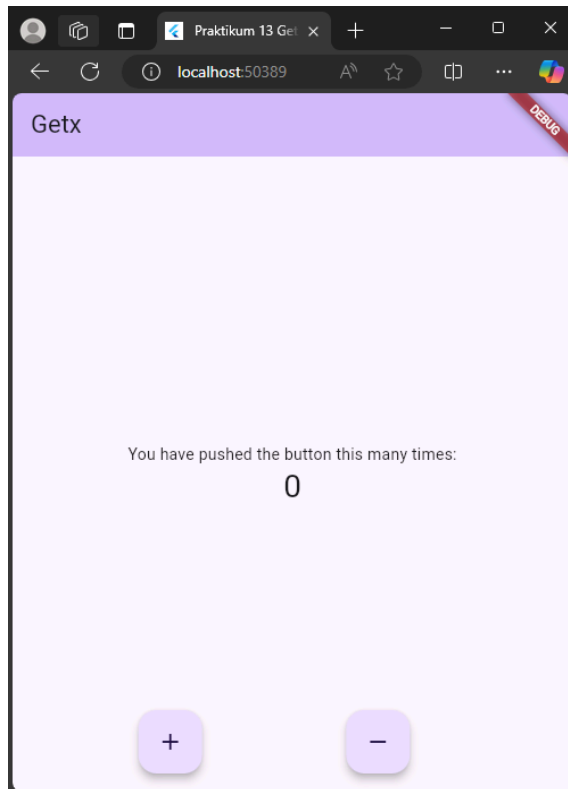
1. Tampilan awal program



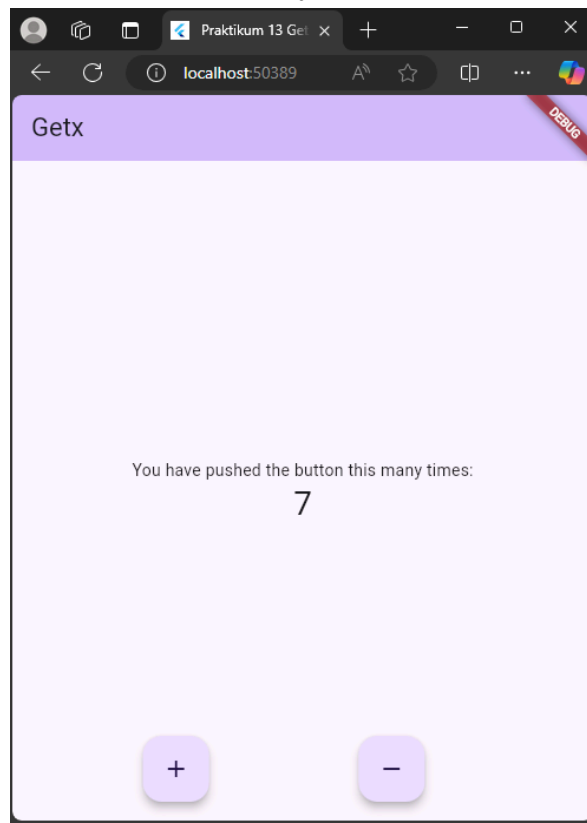
2. Tampilan ketika menekan tombol (+) sebanyak 3 kali



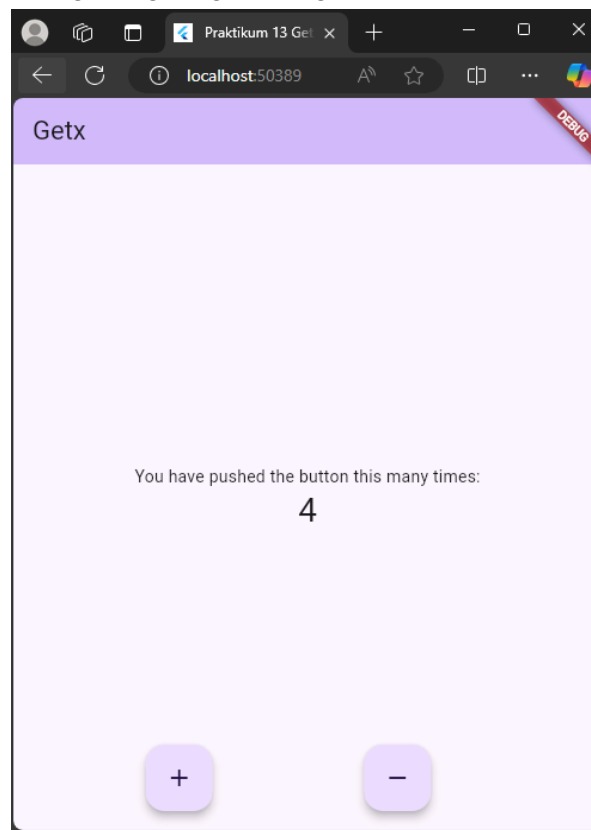
3. Tampilan ketika terdapat tombol (-)



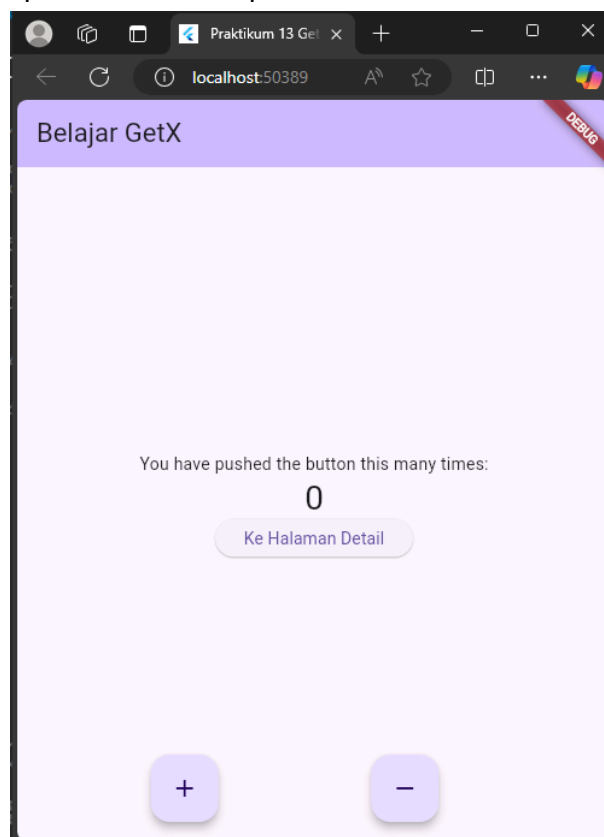
4. Tampilan ketika menekan tombol (+) sebanyak 7 kali



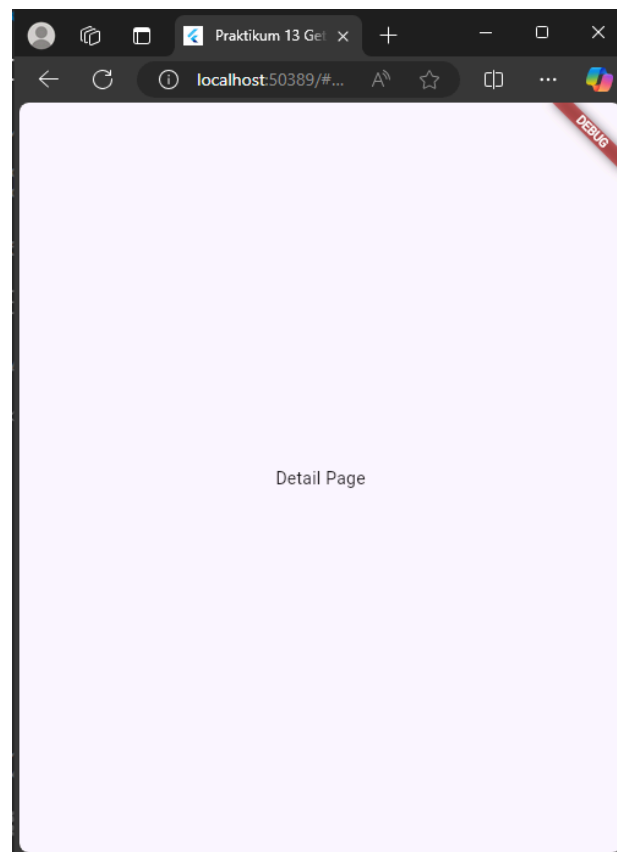
5. Lalu pengguna ingin mengurangi angka dengan menekan tombol (-)



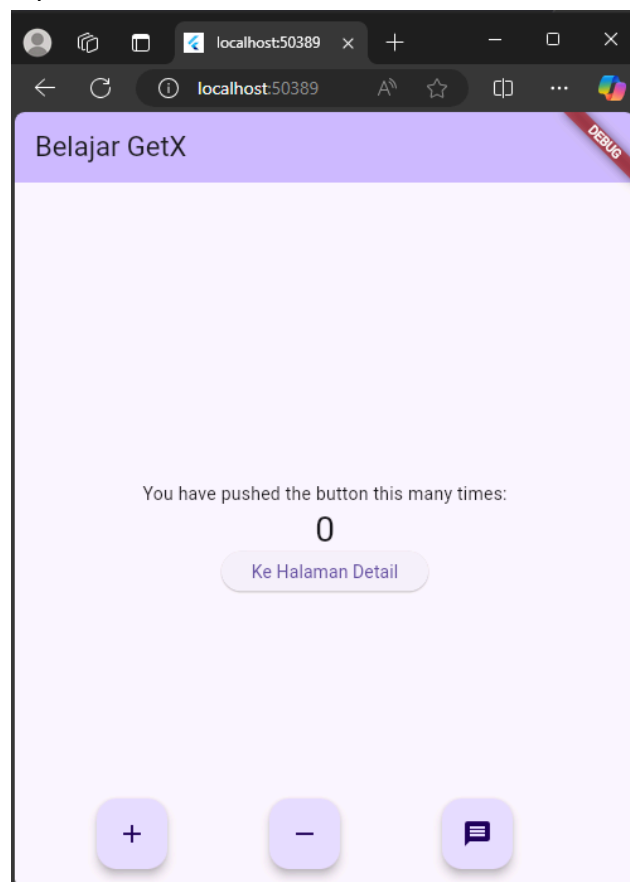
6. Tampilan ketika terdapat tombol untuk pindah ke Halaman Detail



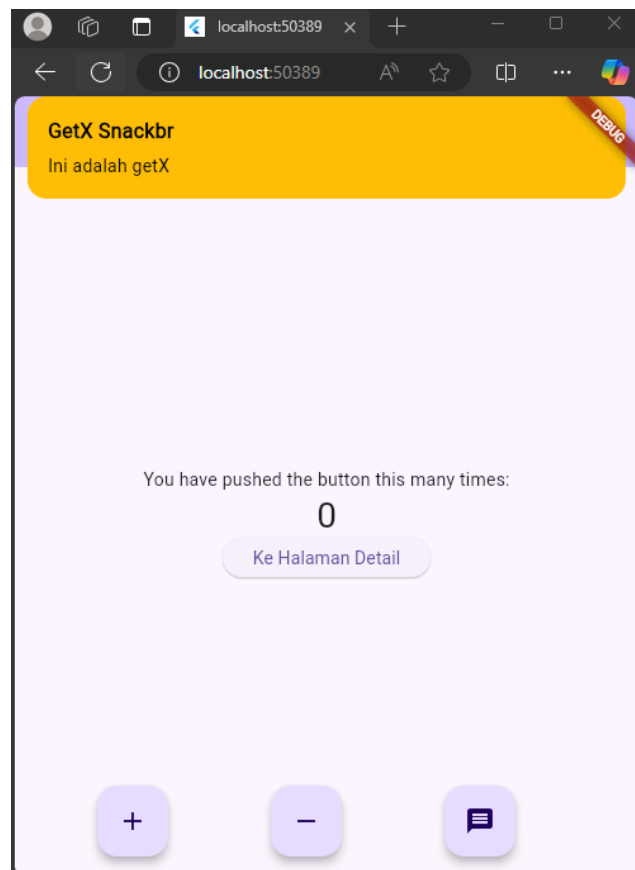
7. Tampilan ketika sudah berpindah ke Halaman Detail



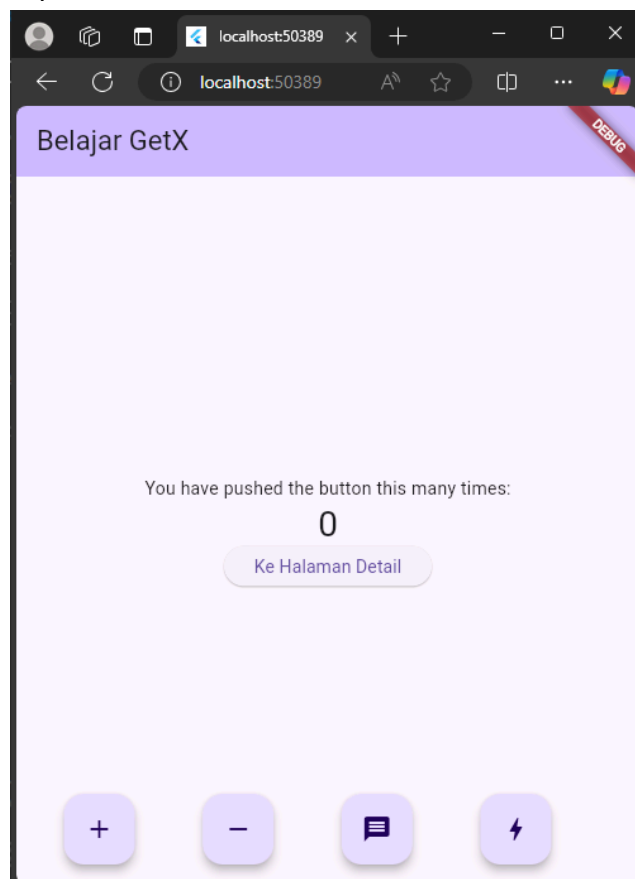
8. Tampilan ketika terdapat tombol Snackbar



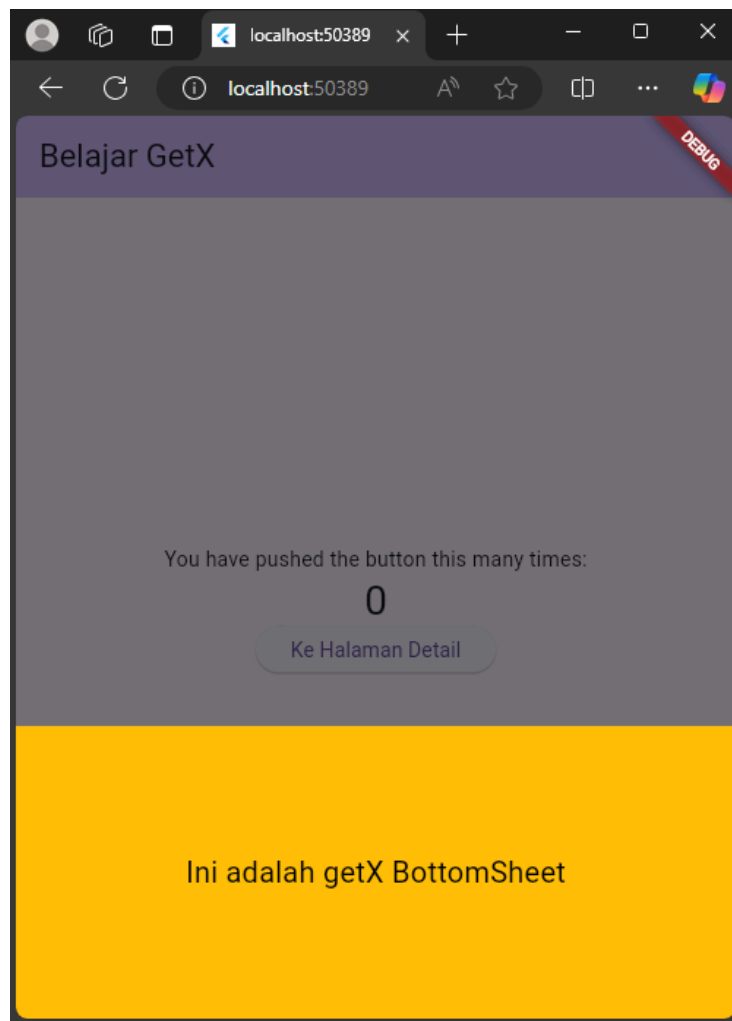
9. Tampilan setelah menekan tombol Snackbar



10. Tampilan ketika terdapat tombol BottomSheet



11. Tampilan setelah menekan tombol BottomSheet.



UNGUIDED

Buatlah Aplikasi Catatan Sederhana menggunakan GetX, dengan ketentuan sebagai berikut :

1. Halaman utama atau Homepage untuk menampilkan daftar catatan yang telah ditambahkan. Setiap catatan terdiri dari judul dan deskripsi singkat, serta terdapat tombol untuk menghapus catatan dari daftar.
2. Halaman kedua untuk menambah catatan baru, berisi : form untuk memasukkan judul dan deskripsi catatan, serta tombol untuk menyimpan catatan ke daftar (Homepage).
3. Menggunakan getx controller.
4. Menggunakan getx routing untuk navigasi halaman

Note: Jangan lupa sertakan source code, screenshot output, dan deskripsi program.

Kreatifitas menjadi nilai tambah

Source Code

main.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul13/view/my_home_page.dart';
import 'package:modul13/view/page_tambah_catatan.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      initialRoute: '/',
      getPages: [
        GetPage(name: '/', page: () => MyHomePage()),
        GetPage(name: '/add', page: () => PageTambahCatatan()),
      ],
    );
  }
}
```

my_home_page.dart

```
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul13/view_model/controller.dart';

class MyHomePage extends StatelessWidget {
  MyHomePage({super.key});

  final NoteController noteController = Get.put(NoteController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Aplikasi Catatan Sederhana'),
        centerTitle: true,
        backgroundColor: const Color.fromARGB(255, 227, 138, 243),
        foregroundColor: Colors.white,
      ),
      body: Obx(
        () => noteController.notes.isEmpty
          ? const Center(
              child: Text('Yahh masih belum ada catatan :('),
            )
          : ListView.builder(
              itemCount: noteController.notes.length,
              itemBuilder: (context, index) {
                final note = noteController.notes[index];
                return Card(
                  child: ListTile(
                    title: Text(note['title']!),
                    subtitle: Text(note['description']!),
                    trailing: IconButton(
                      icon: const Icon(Icons.delete),
                      onPressed: () {
                        noteController.deleteNote(index);
                      },
                    ),
                  ),
                );
              },
            );
      ),
    ),
  ),
);
```

```

floatingActionButton: FloatingActionButton(
  onPressed: () {
    Get.toNamed('/add');
  },
  child: const Icon(Icons.add),
  backgroundColor: const Color.fromARGB(255, 227, 138, 243),
  foregroundColor: Colors.white,
),
);
}
}

```

page_tambah_catatan.dart

```

import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:modul13/view_model/controller.dart';

class PageTambahCatatan extends StatelessWidget {
  PageTambahCatatan({super.key});

  final NoteController noteController = Get.find<NoteController>();
  final TextEditingController titleController = TextEditingController();
  final TextEditingController descriptionController =
TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Tambah Catatan'),
        centerTitle: true,
        backgroundColor: const Color.fromARGB(255, 227, 138, 243),
foregroundColor: Colors.white,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextField(
              controller: titleController,
              decoration: const InputDecoration(

```



```

        labelText: 'Judul',
        border: OutlineInputBorder(),
    ),
),
const SizedBox(height: 16),
TextField(
    controller: descriptionController,
    decoration: const InputDecoration(
        labelText: 'Deskripsi',
        border: OutlineInputBorder(),
    ),
    maxLines: 3,
),
const SizedBox(height: 16),
ElevatedButton(
    onPressed: () {
        if (titleController.text.isNotEmpty &&
            descriptionController.text.isNotEmpty) {
            noteController.addNote(
                titleController.text,
                descriptionController.text,
            );
            Get.back(); // Kembali ke halaman utama
        } else {
            Get.snackbar(
                'Kesalahan',
                'Judul dan Deskripsi tidak boleh kosong',
                backgroundColor: Colors.red,
                colorText: Colors.white,
            );
        }
    },
    child: const Text('Simpan Catatan'),
),
],
),
),
);
}
}

```

controller.dart

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

class NoteController extends GetxController {
  // List untuk menyimpan catatan
  var notes = <Map<String, String>>[].obs;

  // Fungsi untuk menambah catatan
  void addNote(String title, String description) {
    notes.add({'title': title, 'description': description});
  }

  // Fungsi untuk menghapus catatan
  void deleteNote(int index) {
    notes.removeAt(index);
  }
}
```

Output Program

Halaman awal program



Tampilan setelah user menekan tombol (+) untuk menambah catatan



← Tambah Catatan


DEBUG

Judul

Deskripsi

Simpan Catatan

Tampilan setelah pengguna menambahkan catatan



Aplikasi Catatan Sederhana

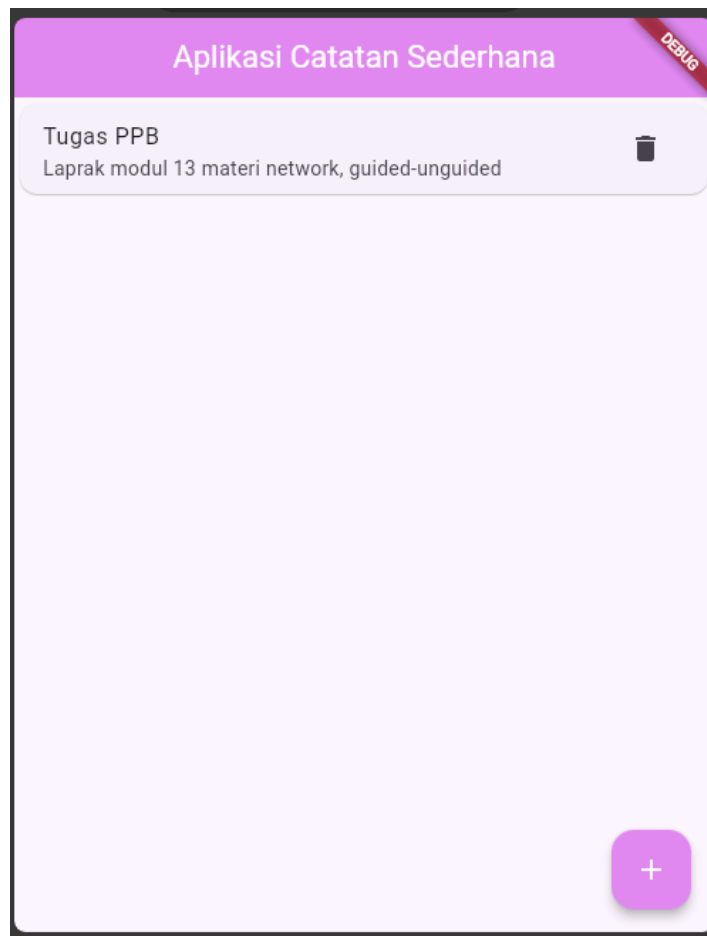
DEBUG

Tugas PPB
Laprak modul 13 materi network, guided-unguided

Rapat panitia
Rapat besar panitia diesnat, di rek 301

+

Tampilan setelah pengguna menghapus catatan “Rapat panitia”



Deskripsi Program:

Program diatas adalah aplikasi catatan sederhana yang menggunakan GetX untuk mengelola state dan navigasi antar halaman. Aplikasi ini mempunyai dua fitur utama yaitu menampilkan daftar catatan di halaman utama dan menambah catatan baru di halaman kedua.

GetX Controller berfungsi untuk mengelola data dan logika aplikasi. Ketika pengguna menambah catatan melalui halaman kedua “PageTambahCatatan” maka akan memanggil fungsi `addNote` untuk menambahkan judul dan deskripsi catatan ke daftar. Fungsi ini memastikan bahwa setiap perubahan pada data akan otomatis diperbarui di tampilan tanpa memerlukan pengaturan ulang manual. Misalnya, jika pengguna menghapus catatan, fungsi `deleteNote` akan langsung memperbarui daftar catatan yang terlihat di halaman utama (MyHomePage).

GetX Routing digunakan untuk mempermudah navigasi antar halaman. Ketika pengguna berada di halaman pertama lalu menekan tombol (+) untuk menambahkan catatan, maka aplikasi akan berpindah ke halaman kedua dengan perintah `Get.toNamed('/add')`. Lalu ketika catatan baru disimpan, perintah `Get.back()` akan membawa pengguna kembali ke halaman utama dan menampilkan data terbaru. Lalu pengguna juga bisa menghapus catatan dengan menekan ikon sampah.