

**TUGAS PENDAHULUAN
PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL XIII
NETWORKING**



Disusun Oleh :

Farhan Kurniawan / 2311104073

SE-07-02

Asisten Praktikum :

Zulfa Mustafa Akhyar Iswahyudi

Yoga Eka Pratama

Dosen Pengampu :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

TUGAS PENDAHULUAN

A. SOAL

1. Apa yang dimaksud dengan state management pada Flutter?
2. Sebut dan jelaskan komponen-komponen yang ada di dalam GetX.
3. Lengkapilah code di bawah ini, dan tampilkan hasil outputnya serta jelaskan.

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

/// Controller untuk mengelola state counter
class CounterController extends GetxController {
  // TODO: Tambahkan variabel untuk menyimpan nilai counter

  // TODO: Buat fungsi untuk menambah nilai counter

  // TODO: Buat fungsi untuk mereset nilai counter
}

class HomePage extends StatelessWidget {
  final CounterController controller =
    Get.put(CounterController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Counter App")),
      body: Center(
        child: Obx(() {
          // TODO: Lengkapi logika untuk menampilkan nilai
          counter
          return Text(
            "0", // Ganti ini dengan nilai counter
            style: TextStyle(fontSize: 48),
          );
        }),
      ),
      floatingActionButton: Column(
```

```

        mainAxisAlignment: MainAxisAlignment.end,
        children: [
          FloatingActionButton(
            onPressed: () {
              // TODO: Tambahkan logika untuk menambah nilai
counter
            },
            child: Icon(Icons.add),
          ),
          SizedBox(height: 10),
          FloatingActionButton(
            onPressed: () {
              // TODO: Tambahkan logika untuk mereset nilai
counter
            },
            child: Icon(Icons.refresh),
          ),
        ],
      ),
    );
  }
}

void main() {
  runApp(MaterialApp(
    debugShowCheckedModeBanner: false,
    home: HomePage(),
  ));
}

```

Screenshoot Output

(lampirkan bukti screenshoot output dari sourcecode)

Deskripsi Program

(deskripsikan program apa yang dibuat, memakai algoritma, dan cara kerja program sampai ke output yang dihasilkan dengan bahasa sendiri) **minimal 5 kalimat.**

B. JAWABAN

1. Pengertian State Management pada Flutter

State management pada Flutter adalah mekanisme untuk mengelola dan mengatur perubahan data (state) dalam sebuah aplikasi sehingga tampilan UI selalu sinkron dengan kondisi data terbaru. State dapat berupa nilai variabel, status login, data hasil request API, maupun interaksi pengguna. Dengan state management yang baik, pengembangan aplikasi menjadi lebih terstruktur, mudah dipelihara, dan efisien karena hanya bagian UI yang membutuhkan saja yang akan diperbarui ketika state berubah.

2. Komponen-komponen pada GetX

Berikut adalah komponen utama yang terdapat pada GetX:

1. State Management

GetX menyediakan state management reaktif menggunakan variabel bertipe Rx (misalnya RxInt, RxString). Perubahan nilai state akan otomatis memperbarui UI tanpa perlu setState().

2. Dependency Management

GetX memudahkan pengelolaan dependensi menggunakan Get.put(), Get.find(), dan Get.lazyPut(). Hal ini membuat kode lebih rapi dan menghindari penggunaan constructor yang berlebihan.

3. Route Management (Navigation)

GetX menyediakan navigasi sederhana tanpa context, seperti Get.to(), Get.back(), dan Get.offAll(), sehingga perpindahan halaman menjadi lebih mudah dan konsisten.

4. Controller (GetxController)

Controller berfungsi untuk menyimpan logika bisnis dan state aplikasi. Dengan memisahkan logika dari UI, aplikasi menjadi lebih terstruktur dan mudah dikembangkan.

3. Source Code Program Counter dengan GetX

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

/// Controller untuk mengelola state counter
class CounterController extends GetxController {
  // Variabel observable untuk menyimpan nilai counter
  // .obs membuat variabel ini reactive (bisa diobservasi
  perubahannya)
  var counter = 0.obs;

  // Fungsi untuk menambah nilai counter
  void increment() {
    counter.value++;
  }

  // Fungsi untuk mereset nilai counter ke 0
  void reset() {
    counter.value = 0;
  }
}

class HomePage extends StatelessWidget {
  // Inisialisasi controller menggunakan Get.put()
  final CounterController controller =
  Get.put(CounterController());

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Counter App"),
        backgroundColor: Colors.blue,
```

```

    ),
    body: Center(
      child: Obx() {
        // Obx() akan rebuild widget ini setiap kali counter berubah
        return Text(
          "${controller.counter.value}", // Menampilkan nilai
counter
          style: TextStyle(fontSize: 48, fontWeight:
FontWeight.bold),
        );
      },
    ),
    floatingActionButton: Column(
      mainAxisAlignment: MainAxisAlignment.end,
      children: [
        FloatingActionButton(
          onPressed: () {
            // Memanggil fungsi increment untuk menambah counter
            controller.increment();
          },
          child: Icon(Icons.add),
          backgroundColor: Colors.green,
          heroTag: "increment", // Menghindari konflik jika ada
multiple FAB
        ),
        SizedBox(height: 10),
        FloatingActionButton(
          onPressed: () {
            // Memanggil fungsi reset untuk mereset counter ke 0
            controller.reset();
          },
          child: Icon(Icons.refresh),
          backgroundColor: Colors.orange,

```

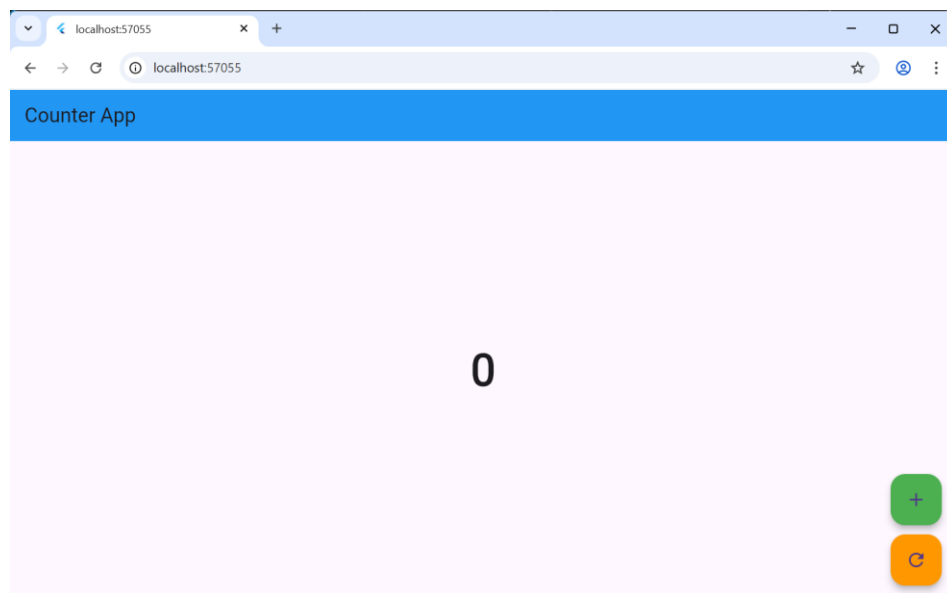
```

        heroTag: "reset", // Menghindari konflik jika ada multiple
      FAB
    ),
  ],
),
);
}
}

void main() {
  runApp(MaterialApp(
    debugShowCheckedModeBanner: false,
    home: HomePage(),
  ));
}

```

Output Program:



Penjelasan:

GetX menggunakan pola State Management yang memisahkan logika bisnis (Controller) dari tampilan (View). Di dalam CounterController, variabel counter dibuat sebagai observable menggunakan .obs yang memungkinkan GetX mendeteksi setiap perubahan nilai secara otomatis. Fungsi increment()

digunakan untuk menambah nilai counter dengan mengakses `counter.value++`, sedangkan fungsi `reset()` mengembalikan nilai counter ke 0. Di `HomePage`, controller diinisialisasi menggunakan `Get.put(CounterController())` yang memasukkan controller ke dalam memory `GetX` sehingga bisa diakses dari mana saja. Widget `Obx()` membungkus `Text` yang menampilkan nilai counter, dan widget ini akan otomatis rebuild setiap kali nilai observable berubah tanpa perlu memanggil `setState()` seperti pada `StatefulWidget` biasa. Kedua `FloatingActionButton` memanggil fungsi `controller.increment()` dan `controller.reset()` untuk mengubah nilai counter, dan perubahan tersebut langsung tercermin di UI secara reaktif berkat sistem observability `GetX`, sehingga membuat state management menjadi lebih sederhana dan efisien dibandingkan dengan pendekatan tradisional Flutter.