TUGAS PENDAHULUAN
PEMROGRAMAN PERANGKAT BERGERAK

MODUL XIII

NETWORKING

Disusun Oleh :

Ilham Lii Assidaq

2311104068

Asisten Praktikum :

Yoga Eka Pratama
Zulfa Mustafa Akhyar Iswahyudi

Dosen Pengampu :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

TUGAS PENDAHULUAN

SOAL

1. Apa yang dimaksud dengan state management pada Flutter?
   State management pada Flutter adalah mekanisme untuk mengelola dan mengontrol perubahan state atau kondisi data dalam sebuah aplikasi sehingga tampilan antarmuka (UI) selalu selaras dengan data yang digunakan. State sendiri merupakan data yang dapat berubah seiring interaksi pengguna atau proses tertentu, seperti nilai counter, status login, maupun data yang diperoleh dari API. Karena Flutter menggunakan pendekatan declarative UI, setiap perubahan state akan memicu proses rebuild pada widget terkait. Oleh karena itu, state management diperlukan agar perubahan data dapat ditangani secara terstruktur, efisien, dan mudah dipelihara, terutama pada aplikasi yang kompleks, sehingga logika aplikasi dan tampilan UI tetap terorganisasi dengan baik.
2. Sebut dan jelaskan komponen-komponen yang ada di dalam GetX.
   GetX memiliki beberapa komponen utama yang saling terintegrasi untuk mempermudah pengembangan aplikasi Flutter, khususnya dalam pengelolaan state, navigasi, dan dependensi. Komponen pertama adalah State Management, yang berfungsi untuk mengelola perubahan state secara reaktif maupun sederhana tanpa memerlukan setState. GetX menyediakan dua pendekatan, yaitu Simple State Manager (GetBuilder) untuk kasus sederhana dan Reactive State Manager (Obx, Rx) untuk state yang membutuhkan pembaruan otomatis saat data berubah. Komponen kedua adalah Route Management (Navigation), yang memungkinkan perpindahan antar halaman tanpa menggunakan BuildContext, sehingga navigasi menjadi lebih ringkas dan mudah, misalnya dengan Get.to() atau Get.back(). Komponen ketiga adalah Dependency Management, yang berfungsi untuk mengatur pembuatan dan penggunaan controller atau service secara efisien melalui fitur dependency injection seperti Get.put(), Get.find(), dan Get.lazyPut(). Selain itu, GetX juga menyediakan Utilities seperti snackbar, dialog, dan bottom sheet yang dapat dipanggil secara global tanpa context, sehingga mempercepat proses pengembangan aplikasi Flutter secara keseluruhan.
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(
```

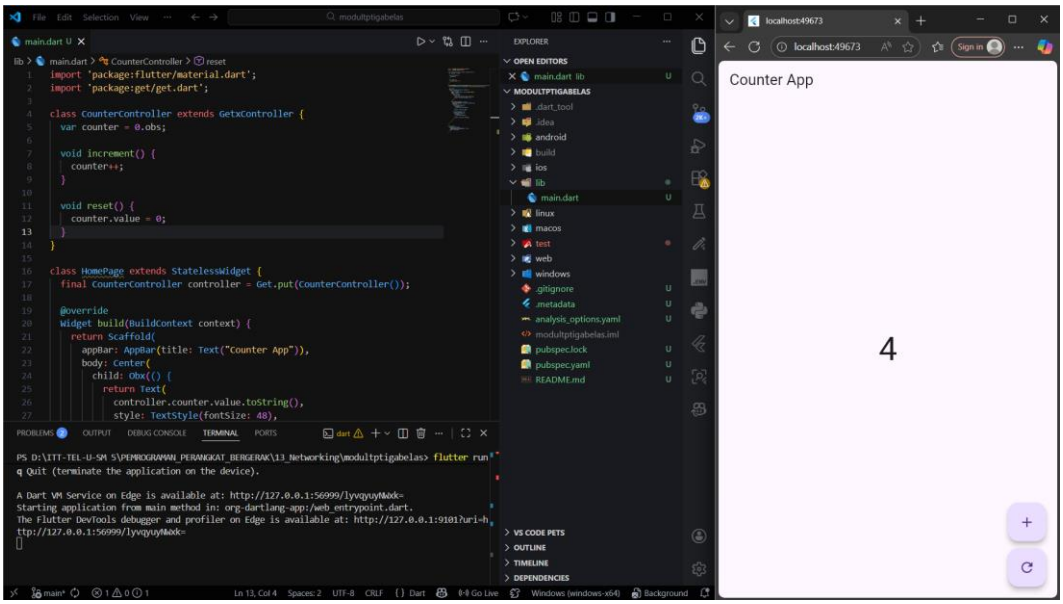
```
            onPressed: () {
              // TODO: Tambahkan logika untuk mereset nilai
counter
            },
            child: Icon(Icons.refresh),
          ),
        ],
      ),
    );
  }
}

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

Screenshoot Output

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

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

 void increment() {
  counter++;
 }

 void reset() {
  counter.value = 0;
 }
}

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(() {
      return Text(
       controller.counter.value.toString(),
       style: TextStyle(fontSize: 48),
      );
     }),
    ),
    floatingActionButton: Column(
     mainAxisAlignment: MainAxisAlignment.end,
     children: [
      FloatingActionButton(
       onPressed: () {
        controller.increment();
       },
       child: Icon(Icons.add),
      ),
      SizedBox(height: 10),
      FloatingActionButton(
       onPressed: () {
        controller.reset();
       },
       child: Icon(Icons.refresh),
      ),
     ],
    ),
   );
  }
}
```

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



Deskripsi Program

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

Program ini merupakan aplikasi Counter sederhana berbasis Flutter yang menerapkan state management menggunakan GetX. Algoritma program dimulai dengan membuat sebuah controller (CounterController) yang menyimpan nilai counter dalam bentuk variabel reaktif (counter) dan menyediakan dua operasi utama, yaitu menambah nilai counter dan mengatur ulang nilai counter ke nol.

Ketika aplikasi dijalankan, HomePage akan menginisialisasi controller menggunakan Get.put, sehingga state counter dapat diakses dan dikelola secara terpusat.

Tampilan utama aplikasi menampilkan nilai counter di tengah layar, dan nilai ini akan otomatis diperbarui setiap kali terjadi perubahan karena dibungkus dengan widget Obx.

Saat pengguna menekan tombol tambah (+), algoritma akan menaikkan nilai counter satu per satu, sedangkan saat tombol reset ditekan, nilai counter dikembalikan ke nol, dan hasil akhirnya langsung ditampilkan pada layar tanpa perlu melakukan refresh manual.