**TUGAS PENDAHULUAN**

**PEMROGRAMAN PERANGKAT BERGERAK**

**MODUL XIII**

**NETWORKING**



**Disusun Oleh :**

**Fadhila Agil Permana / 2211104006**

**SE-06-1**

**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**

**TUGAS PENDAHULUAN**

**SOAL**

1. Apa yang dimaksud dengan state management pada Flutter?
   1. **State management** dalam Flutter merujuk pada cara kita mengelola dan memperbarui status (state) aplikasi secara efisien. State adalah informasi yang dapat berubah selama aplikasi berjalan, seperti data input dari pengguna, status tampilan, atau data yang diambil dari API. Ada banyak pendekatan untuk mengelola state dalam Flutter, seperti menggunakan **Provider**, **Riverpod**, **Bloc**, dan **GetX**. Setiap pendekatan memiliki kelebihan dan kekurangan tergantung pada kompleksitas aplikasi.
2. Sebut dan jelaskan komponen-komponen yang ada di dalam GetX.
   1. GetX adalah paket Flutter yang menawarkan **state management** dan **dependency injection** yang sederhana dan efisien. Komponen utama di dalam GetX adalah:
   2. **GetxController**: Ini adalah kelas yang digunakan untuk mengelola logika bisnis dan state dalam aplikasi. Anda dapat membuat controller yang berisi variabel yang dapat di-observe (seperti Rx<T>), serta fungsi untuk memperbarui state tersebut.
   3. **Rx (Reactive)**: Variabel reaktif yang memungkinkan perubahan nilai secara otomatis diberitahukan kepada widget yang menggunakannya. Contohnya adalah RxInt, RxString, dll.
   4. **Get.put()**: Digunakan untuk menyimpan instance dari controller sehingga bisa diakses di berbagai bagian aplikasi.
   5. **Obx()**: Widget yang digunakan untuk memantau perubahan state. Widget ini akan membangun ulang tampilan ketika nilai dalam controller berubah.
   6. **GetX()**: Widget lain yang mirip dengan Obx(), tetapi dengan lebih banyak fleksibilitas.
   7. **GetBuilder()**: Widget yang juga digunakan untuk pembaruan UI, tetapi berbeda dengan Obx() karena GetBuilder() tidak membutuhkan Rx dan lebih cocok untuk mengelola state yang tidak berubah sering.
   8. **Get.lazyPut()**: Digunakan untuk melakukan lazy initialization pada controller, yaitu controller hanya dibuat saat pertama kali dibutuhkan
3. Lengkapilah code di bawah ini, dan tampilkan hasil outputnya serta jelaskan.

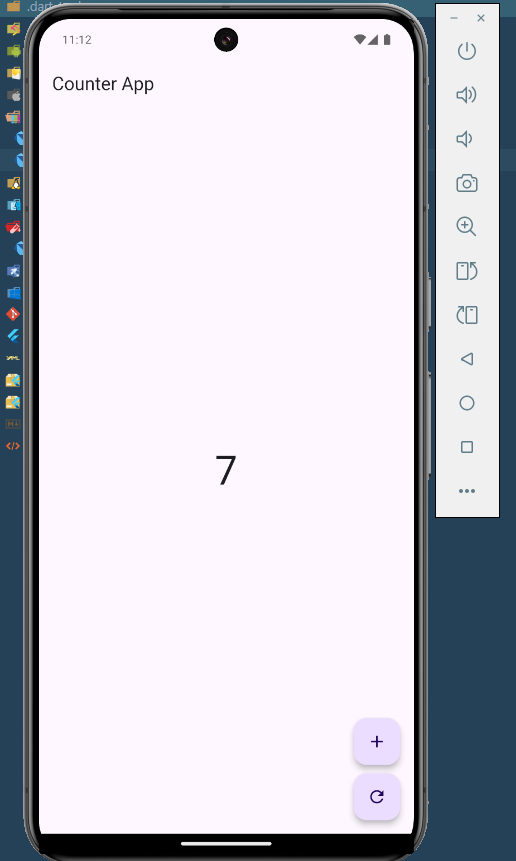
|  |
| --- |
| import 'package:flutter/material.dart'; import 'package:get/get.dart';    /// Controller untuk mengelola state counter class CounterController extends GetxController {  // TODO: Tambahkan variabel untuk menyimpan nilai counter  // TODO: Buat fungsi untuk menambah nilai counter    // TODO: 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(),  ));  } |

**KODE YANG DILEKAPKAN**

|  |
| --- |
| import 'package:flutter/material.dart';  import 'package:get/get.dart';  */// Controller untuk mengelola state counter*  class CounterController extends GetxController {  *// Variabel untuk menyimpan nilai counter*    var **counter** = 0.**obs**;  *// Fungsi untuk menambah nilai counter*    void increment() {  **counter**++;    }  *// Fungsi untuk mereset nilai counter*    void reset() {  **counter**.**value** = 0;    }  }  class HomePage extends StatelessWidget {    final CounterController **controller** = **Get**.put(CounterController());    HomePage({super.key});    @**override**    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(title: const Text("Counter App")),        body: Center(          child: Obx(() {  *// Menampilkan nilai counter*            return Text(              "${**controller**.**counter**}",              style: const TextStyle(fontSize: 48),            );          }),        ),        floatingActionButton: Column(          mainAxisAlignment: MainAxisAlignment.end,          children: [            FloatingActionButton(              onPressed: () {  *// Menambah nilai counter*  **controller**.increment();              },              child: const Icon(Icons.**add**),            ),            const SizedBox(height: 10),            FloatingActionButton(              onPressed: () {  *// Mereset nilai counter*  **controller**.reset();              },              child: const 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.**