**Ketentuan;**

Simpanlah file lembar jawaban ini dengan format; **Kelas\_Nama Lengkap**

Contoh; **TI 3C\_Rifki Fakhrudin**

**Upload tugas jobsheet ini dengan batas maksimum sesuai jadwal perkuliahan masing masing kelas**

Upload file tugas jobsheet di website **lms.polinema**

|  |  |  |
| --- | --- | --- |
| **Nama** | : | Rifki Fakhrudin |
| **Nim** | : | 2241720218 |
| **Kelas** | : | 3C |

Tulislah Jawaban Pada Kolom Yang tersedia di bawah ini;

**LEMBAR JAWABAN JOBSHEET-12**

|  |  |
| --- | --- |
| **Soal No** | **Jawaban** |
|  | **Praktikum 1**  **Langkah 1: Buat Project Baru** |
|  | **Langkah 2: Buka file main.dart**  import 'package:flutter/material.dart';  void main() {    runApp(const MyApp());  }  class MyApp extends StatelessWidget {    const MyApp({super.key});    @override    Widget build(BuildContext context) {      return MaterialApp(        title: 'Stream',        theme: ThemeData(          primarySwatch: Colors.deepPurple,        ),        home: const StreamHomePage(),      );    }  }  class StreamHomePage extends StatefulWidget {    const StreamHomePage({super.key});    @override    State<StreamHomePage> createState() => \_StreamHomePageState();  }  class \_StreamHomePageState extends State<StreamHomePage> {    @override    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(          title: const Text('Stream Home Page'),        ),      );    }  }  **Soal 1**   * **Tambahkan nama panggilan Anda pada title app sebagai identitas hasil pekerjaan Anda.**      * **Gantilah warna tema aplikasi sesuai kesukaan Anda.**      * **Lakukan commit hasil jawaban Soal 1 dengan pesan "W13: Jawaban Soal 1"** |
|  | **Langkah 3: Buat file baru stream.dart** |
|  | **Langkah 4: Tambah variabel colors**    **Soal 2**   * **Tambahkan 5 warna lainnya sesuai keinginan Anda pada variabel colors tersebut.**      * **Lakukan commit hasil jawaban Soal 2 dengan pesan "W13: Jawaban Soal 2"** |
|  | **Langkah 5: Tambah method getColors()** |
|  | **Langkah 6: Tambah perintah yield\***    **Soal 3**   * **Jelaskan fungsi keyword yield\* pada kode tersebut!** * Fungsi keyword yield\* pada kode tersebut digunakan untuk "mengalirkan" nilai dari sebuah stream atau iterable ke dalam generator yang sedang berjalan. * **Apa maksud isi perintah kode tersebut?**   + Kode ini menghasilkan stream yang mengeluarkan warna secara periodik setiap detik, dengan indeks warna yang berulang berdasarkan waktu detik (t). * **Lakukan commit hasil jawaban Soal 3 dengan pesan "W13: Jawaban Soal 3"** |
|  | **Langkah 7: Buka main.dart** |
|  | **Langkah 8: Tambah variabel** |
|  | **Langkah 9: Tambah method changeColor()** |
|  | **Langkah 10: Lakukan override initState()** |
|  | **Langkah 11: Ubah isi Scaffold()** |
|  | **Langkah 12: Run**    **Soal 4**   * **Capture hasil praktikum Anda berupa GIF dan lampirkan di README.**   + **Background diatas bisa berganti warna dengan warna dari class stream.dart** * **Lakukan commit hasil jawaban Soal 4 dengan pesan "W13: Jawaban Soal 4"** |
|  | **Langkah 13: Ganti isi method changeColor()**    **Soal 5**   * **Jelaskan perbedaan menggunakan listen dan await for (langkah 9) !**   + Kode 1 (await for) akan menunggu stream selesai sebelum melanjutkan eksekusi kode berikutnya. Hal ini lebih cocok untuk situasi di mana Anda ingin memproses setiap nilai stream secara berurutan dan menunggu semua data diterima.   + Kode 2 (listen()) memungkinkan stream untuk tetap berjalan dan mendengarkan nilai baru yang dipancarkan tanpa menunggu stream selesai. Ini lebih cocok untuk aliran data yang berlangsung terus-menerus. * **Lakukan commit hasil jawaban Soal 5 dengan pesan "W13: Jawaban Soal 5"** |
|  | **Praktikum 2**  **Langkah 1: Buka file stream.dart** |
|  | **Langkah 2: Tambah class NumberStream** |
|  | **Langkah 3: Tambah StreamController** |
|  | **Langkah 4: Tambah method addNumberToSink** |
|  | **Langkah 5: Tambah method close()** |
|  | **Langkah 6: Buka main.dart** |
|  | **Langkah 7: Tambah variabel** |
|  | **Langkah 8: Edit initState()** |
|  | **Langkah 9: Edit dispose()** |
|  | **Langkah 10: Tambah method addRandomNumber() void addRandomNumber() {** |
|  | **Langkah 11: Edit method build()** |
|  | **Langkah 12: Run**    **Soal 6**   * **Jelaskan maksud kode langkah 8 dan 10 tersebut!**   + Langkah 8 : berfungsi untuk mendengarkan dan memproses nilai stream   + Langkah 10: menghasilkan dan mengirimkan nilai acak ke dalam stream. * **Capture hasil praktikum Anda berupa GIF dan lampirkan di README.**      * **Lalu lakukan commit dengan pesan "W13: Jawaban Soal 6".** |
|  | **Langkah 13: Buka stream.dart** |
|  | **Langkah 14: Buka main.dart** |
|  | **Langkah 15: Edit method addRandomNumber()**     * **Jelaskan maksud kode langkah 13 sampai 15 tersebut!**   + Langkah 13 : menambah error ke dalam stream secara eksplisit.   + Langkah 14: mendengarkan stream dan mengatur nilai berdasarkan data atau error yang diterima.   + Langkah 15: menghasilkan angka acak dan kemudian memicu error ke dalam stream menggunakan addError(). * **Kembalikan kode seperti semula pada Langkah 15, comment addError() agar Anda dapat melanjutkan ke praktikum 3 berikutnya.**      * **Lalu lakukan commit dengan pesan "W13: Jawaban Soal 7".** |
|  | **Praktikum 3**  **Langkah 1: Buka main.dart** |
|  | **Langkah 2: Tambahkan kode ini di initState** |
|  | **Langkah 3: Tetap di initState** |
|  | **Langkah 4: Run**     * **Jelaskan maksud kode langkah 1-3 tersebut!**   + Langkah 1 : mendeklarasikan variabel transformer yang akan digunakan untuk mentransformasi data dalam stream.   + Langkah 2 : menginisialisasi transformer untuk mengubah nilai stream (mengalikan nilai dengan 10) dan menangani error serta menyelesaikan stream.   + Langkah 3 : mengaplikasikan transformer pada stream yang diterima dari numberStreamController, mendengarkan perubahan data, mengupdate UI dengan data baru atau error, dan juga menginisialisasi stream warna untuk memperbarui latar belakang. * **Capture hasil praktikum Anda berupa GIF dan lampirkan di README.**      * **Lalu lakukan commit dengan pesan "W13: Jawaban Soal 8".** |
|  | **Praktikum 4**  **Langkah 1: Tambah variabel** |
|  | **Langkah 2: Edit initState()** |
|  | **Langkah 3: Tetap di initState()** |
|  | **Langkah 4: Tambah properti onDone()** |
|  | **Langkah 5: Tambah method baru** |
|  | **Langkah 6: Pindah ke method dispose()** |
|  | **Langkah 7: Pindah ke method build()** |
|  | **Langkah 8: Edit method addRandomNumber()** |
|  | **Langkah 9: Run** |
|  | **Langkah 10: Tekan button ‘Stop Subscription'**    **Soal 9**   * **Jelaskan maksud kode langkah 2, 6 dan 8 tersebut!**   + Langkah 2 : Mendaftar ke stream dan menangani event yang diterima;   + Langkah 6: Membatalkan (menghentikan) subscription atau pendaftaran ke stream.   + Langkah 8 : Menambahkan angka acak ke dalam stream dan memperbarui UI berdasarkan kondisi apakah stream terbuka atau tertutup. * **Capture hasil praktikum Anda berupa GIF dan lampirkan di README.** * **Lalu lakukan commit dengan pesan "W13: Jawaban Soal 9".** |
|  | **Praktikum 5**  **Langkah 1: Buka file main.dart** |
|  | **Langkah 2: Edit initState()** |
|  | **Langkah 3: Run**  **Soal 10**   * **Jelaskan mengapa error itu bisa terjadi ?**   + Hal itu terjadi karena mencoba untuk mendengarkan (listen) sebuah stream yang sudah didengarkan sebelumnya. |
|  | **Langkah 4: Set broadcast stream** |
|  | **Langkah 5: Edit method build()** |
|  | **Langkah 6: Run**    **Soal 11**   * **Jelaskan mengapa hal itu bisa terjadi ?**   + Hal itu terjadi karena menggunakan Multiple StreamSubscription stream tersebut menjadi sebuah broadcast stream, yang artinya Anda dapat memiliki lebih dari satu pendengar (subscriber) yang mendengarkan stream yang sama secara bersamaan. * **Capture hasil praktikum Anda berupa GIF dan lampirkan di README.** * **Lalu lakukan commit dengan pesan "W13: Jawaban Soal 10,11".** |
|  | **Praktikum 06**  **Langkah 1: Buat Project Baru** |
|  | **Langkah 2: Buat file baru stream.dart** |
|  | **Langkah 3: Tetap di file stream.dart** |
|  | **Langkah 4: Edit main.dart**  import 'package:flutter/material.dart';  import 'stream.dart'; // Ensure the correct file path for this import  import 'dart:async';  void main() {    runApp(const MyApp());  }  class MyApp extends StatelessWidget {    const MyApp({super.key});    @override    Widget build(BuildContext context) {      return MaterialApp(        title: 'Stream',        theme: ThemeData(          primarySwatch: Colors.deepPurple,        ),        home: const StreamHomePage(),      );    }  }  class StreamHomePage extends StatefulWidget {    const StreamHomePage({super.key});    @override    State<StreamHomePage> createState() => \_StreamHomePageState();  }  class \_StreamHomePageState extends State<StreamHomePage> {    @override    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(          title: const Text('stream'),        ),        body: Container(        ),      );    }  } |
|  | **Langkah 5: Tambah variabel** |
|  | **Langkah 6: Edit initState()** |
|  | **Langkah 7: Edit method build()**  body: StreamBuilder(          stream: numberStream, // Gunakan camelCase di sini          initialData: 0,          builder: (context, snapshot) {            if (snapshot.hasError) {              print('Error');            }            if (snapshot.hasData) {              return Center(                child: Text(                  snapshot.data.toString(),                  style: const TextStyle(fontSize: 96),                ),              );            } else {              return const SizedBox.shrink();            }          },        ),      );    }  } |
|  | **Langkah 8: Run**    **Soal 12**   * **Jelaskan maksud kode pada langkah 3 dan 7 !**   + Langkah 3 : membuat fungsi getNumbers(), dan menampilkan nilai baru setiap 1 detik, objek Random menghasilkan nilai baru setiap 1 detik, dan di return menjadi elemen stream.   + Langkah 7: kode tersebut berfungsi untuk menampilkan fungsi dari getNumbers, dan jika terjadi eror maka akan muncul pesan eror. * **Capture hasil praktikum Anda berupa GIF dan lampirkan di README.** * **Lalu lakukan commit dengan pesan "W13: Jawaban Soal 12".** |
|  | **Praktikum 7**  **Langkah 1: Buat Project baru** |
|  | **Langkah 2: Isi kode random\_bloc.dart** |
|  | **Langkah 3: Buat class RandomNumberBloc()** |
|  | **Langkah 4: Buat variabel StreamController** |
|  | **Langkah 5: Buat constructor** |
|  | **Langkah 6: Buat method dispose()** |
|  | **Langkah 7: Edit main.dart** |
|  | **Langkah 8: Buat file baru random\_screen.dart** |
|  | **Langkah 9: Lakukan impor material dan random\_bloc.dart** |
|  | **Langkah 10: Buat StatefulWidget RandomScreen**  import 'package:flutter/material.dart';  import 'package:bloc\_random\_rifki/RandomNumberBloc.dart';  class RandomScreen extends StatefulWidget {    @override    \_RandomScreenState createState() => \_RandomScreenState();  }  class \_RandomScreenState extends State<RandomScreen> {    @override    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(          title: Text("Random Screen"),        ),        body: Center(          child: Column(            mainAxisAlignment: MainAxisAlignment.center,            children: [              // Tambahkan widget di sini            ],          ),        ),      );    }  } |
|  | **Langkah 11: Buat variabel** |
|  | **Langkah 12: Buat method dispose()** |
|  | **Langkah 13: Edit method build()**    **Soal 13**   * **Jelaskan maksud praktikum ini ! Dimanakah letak konsep pola BLoC-nya ?**   + Pola Blo Cnya, untuk logika bisnis sendiri terdapat di kelas Randomnumberbloc. Semua proses pembuatan bilangan acak dilakukan di sini.dan **UI (User Interface)** berada di widget RandomScreen, yang hanya bertugas menampilkan data dari Stream tanpa mengetahui bagaimana data tersebut dihasilkan. * **Capture hasil praktikum Anda berupa GIF dan lampirkan di README.** * **Lalu lakukan commit dengan pesan "W13: Jawaban Soal 13".** |