LAPORAN TEORI MOBILE PROGRAMMING
MODUL 9

Nama : Erai Bagusalim

NIM : 240605110088

Kelas : Mobile Programming B

Tanggal : 6 September 2025

JURUSAN TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM

MALANG

GANJIL 2025/2026

i. **Tujuan**

1. Memahami konsep StatefulWidget dalam Flutter.
2. Membedakan StatelessWidget dan StatefulWidget.
3. Menggunakan fungsi setState() untuk mengubah tampilan secara dinamis.
4. Mengimplementasikan perubahan state sederhana seperti counter dan tombol like/unlike.

ii. **Langkah Kerja**

1. Membuat proyek Flutter baru bernama tasbih_app.
2. Menambahkan dependency simple_circular_progress_bar pada pubspec.yaml.
3. Membuat Stateful Widget MyApp di main.dart.
4. Menambahkan variabel state _valueNotifier dan counter.
5. Menulis method incrementCounter() dan resetCounter() untuk mengubah nilai counter dan circular progres bar.
6. Menyusun tampilan UI pada method build() menggunakan Text, SimpleCircularProgressBar, InkWell, dan FloatingActionButton.
7. Menjalankan aplikasi dan mengecek interaksi tombol tambah dan reset.

iii. **Screenshot Hasil**

a. Kode Program

```dart
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:simple_circular_progress_bar/simple_circular_progress_bar.dart';

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

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

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  late ValueNotifier<double> _valueNotifier;
  late double counter;

  @override
  void initState() {
    super.initState();
    _valueNotifier = ValueNotifier(0.0);
    counter = 0.0;
  }

  @override
  void dispose() {
    _valueNotifier.dispose();
    super.dispose();
  }

  void incrementCounter() {
    setState(() {
      if (counter < 33) {
        counter++;
        _valueNotifier.value = (counter / 33) * 100;
      }
    });
  }

  void resetCounter() {
    setState(() {
      counter = 0.0;
      _valueNotifier.value = (counter / 33) * 100;
    });
  }

  @override
  Widget build(BuildContext context) {
    SystemChrome.setSystemUIOverlayStyle(
      const SystemUiOverlayStyle(statusBarColor: Colors.transparent),
    );
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
          seedColor: const Color.fromARGB(255, 119, 210, 145),
        ), // ColorScheme.fromSeed
        useMaterial3: true,
      ), // ThemeData
      home: Scaffold(
        backgroundColor: const Color.fromARGB(255, 119, 210, 145),
        body: SafeArea(
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  '${(counter.round())}',
                  style: const TextStyle(fontSize: 50),
                ), // Text
                SimpleCircularProgressBar(
                  progressColors: [Colors.amberAccent.shade400],
                  size: 300,
                  progressStrokeWidth: 20,
                  backStrokeWidth: 10,
                  mergeMode: true,
                  maxValue: 100,
                  animationDuration: 0,
                  valueNotifier: _valueNotifier,
                  onGetText: (value) {
                    return Text(
                      '${(value.toInt() / 3).round()}',
                      style: const TextStyle(fontSize: 170),
                    ); // Text
                  },
                ), // SimpleCircularProgressBar
                const SizedBox(height: 50),
                ClipRRect(
                  borderRadius: const BorderRadius.all(Radius.circular(50)),
                  child: InkWell(
                    onTap: incrementCounter,
                    child: Container(
                      decoration: const BoxDecoration(color: Colors.white),
                      child: const Icon(Icons.fingerprint, size: 125),
                    ), // Container
                  ), // InkWell
                ), // ClipRRect
              ],
            ), // Column
          ), // Center
        ), // SafeArea
        floatingActionButton: FloatingActionButton(
          onPressed: resetCounter,
          child: const Icon(Icons.refresh_outlined),
        ), // FloatingActionButton
      ), // Scaffold
    ); // MaterialApp
  }
}
```

b. Penjelasan Kode Program

Aplikasi tasbih digital ini dibangun menggunakan StatefulWidget, yang memungkinkannya untuk mengubah tampilan layar saat pengguna berinteraksi. Inti dari aplikasi ini terletak pada dua variabel utama: counter yang berfungsi sebagai penyimpan nilai hitungan zikir, dan _valueNotifier yang secara khusus mengontrol animasi pada progress bar yang melingkar. Saat aplikasi pertama kali dijalankan, fungsi initState() akan mengatur nilai awal counter menjadi nol. Logika utamanya

ada pada dua fungsi, yaitu incrementCounter() yang akan menambah hitungan setiap kali tombol disentuh (hingga batas 33) dan resetCounter() untuk mengembalikan hitungan ke nol. Keduanya dibungkus dalam setState(), sebuah perintah penting yang memberitahu Flutter untuk segera memperbarui tampilan setiap kali ada perubahan nilai. Tampilan antarmukanya sendiri disusun secara vertikal menggunakan Column, yang berisi teks untuk angka hitungan, widget SimpleCircularProgressBar dari package eksternal, dan tombol utama berupa InkWell dengan ikon sidik jari. Di bagian pojok bawah, terdapat FloatingActionButton yang berfungsi sebagai tombol reset.

Output :

iv. **Kesimpulan**

Secara keseluruhan, proyek ini berhasil menunjukkan bagaimana cara kerja StatefulWidget dalam membuat aplikasi yang dinamis dan interaktif. Kunci utamanya adalah penggunaan setState(), yang terbukti menjadi jembatan antara logika program dan tampilan visual, memastikan setiap perubahan data langsung terlihat oleh pengguna. Dengan mengintegrasikan package dari luar seperti simple_circular_progress_bar, tampilan aplikasi menjadi lebih menarik secara visual dan memberikan feedback progres yang jelas. Penggunaan berbagai jenis tombol seperti InkWell dan FloatingActionButton juga menunjukkan fleksibilitas Flutter dalam menangani input dari pengguna. Pada akhirnya, aplikasi ini adalah contoh nyata dari konsep dasar manajemen state: interaksi pengguna mengubah data (state), dan perubahan data tersebut secara otomatis membangun ulang antarmuka pengguna (UI).