LAPORAN TEORI MOBILE PROGRAMMING
MODUL 9

Nama : Firman Fadilah Noor
NIM : 240605110083
Kelas : B
Tanggal : 6 Oktober 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';

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(225, 119, 210, 145),
        ), // ColorScheme.fromSeed ColorScheme.fromSeed
        useMaterial3: true,
      ), // ThemeData ThemeData
      home: Scaffold(
        backgroundColor: const Color.fromARGB(244, 119, 210, 145),
        body: SafeArea(
          child: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  '${(counter.round())}',
                  style: const TextStyle(fontSize: 50),
                ), // Text Text
                SimpleCircularProgressBar(
                  progressColors: [Colors.amberAccent.shade400],
                  size: 300,
                  progressStrokeWidth: 20,
                  backStrokeWidth: 10,
                  mergeMode: true,
                  maxValue: 100,
                  animationDuration: 0,
                  valueNotifier: _valueNotifier,
                  onGetText: (value) {
                    return Text(
                      '${(value.toInt() / 3).round()}',
                      style: const TextStyle(fontSize: 170),
                    ); // tEXT Text
                  },
                ), // SimpleCircularProgressBar 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 Container
                  ), // InkWell InkWell
                ), // ClipRRect ClipRRect
              ],
            ), // Column Column
          ), // Center Center
        ), // SafeArea SafeArea
        floatingActionButton: FloatingActionButton(
          onPressed: resetCounter,
          child: const Icon(Icons.refresh_outlined),
        ), // FloatingActionButton FloatingActionButton
      ), // Scaffold Scaffold
    ); // MaterialApp MaterialApp
  }
}
```

b. Penjelasan Kode Program.

Kode Flutter di atas sebenarnya membuat aplikasi kecil yang menampilkan lingkaran progres animasi yang terus bertambah setiap kali pengguna menekan tombol bergambar sidik jari, lalu bisa diatur ulang dengan menekan tombol refresh di pojok bawah.

Aplikasi dimulai dari fungsi main() yang menjalankan MyApp(). Karena nilai progres akan berubah setiap kali tombol ditekan, maka digunakan StatefulWidget supaya tampilan bisa diperbarui secara dinamis. Di dalam MyAppState, ada dua variabel utama: _valueNotifier dan counter. Keduanya berfungsi untuk mencatat dan mengatur nilai kemajuan yang akan ditampilkan pada progress bar.

Saat aplikasi baru dijalankan, nilai progres masih nol (0.0). Ketika tombol sidik jari ditekan, fungsi incrementCounter() dipanggil. Fungsi ini menambah nilai counter satu per satu hingga mencapai batas maksimal 33. Setiap kali nilainya berubah, progress bar lingkaran ikut bergerak naik sesuai perhitungan (counter / 33) * 100.

Jika pengguna ingin mengulang dari awal, tombol Floating Action Button dengan ikon refresh dapat ditekan. Tombol ini menjalankan fungsi resetCounter(), yang akan mengembalikan nilai progres ke nol dan memulai ulang animasinya.
Tampilan aplikasinya dibuat dengan gaya modern menggunakan warna lembut dan animasi halus. Teks di tengah menampilkan angka dari progres yang sedang berjalan, sementara lingkaran di sekitarnya menunjukkan visual pergerakan progres tersebut.

Secara sederhana, aplikasi ini menggambarkan bagaimana interaksi pengguna bisa memengaruhi tampilan animasi secara real-time di Flutter — sebuah contoh bagus untuk memahami konsep state management, animasi, dan event handling dalam pemrograman antarmuka Flutter.

Output:

IV. Kesimpulan.

Dari hasil praktikum Modul 9 tentang StatefulWidget pada Flutter, dapat disimpulkan bahwa penggunaan StatefulWidget sangat penting ketika aplikasi membutuhkan tampilan yang bisa berubah sesuai interaksi pengguna. Melalui percobaan ini, kita belajar bagaimana cara kerja state management di Flutter, terutama penggunaan ValueNotifier atau setState() untuk memperbarui tampilan secara langsung.

Aplikasi yang dibuat berupa tasbih digital dengan lingkaran progres animasi, di mana setiap kali tombol ditekan, nilai progres bertambah dan divisualisasikan dengan animasi yang halus. Fitur reset juga menunjukkan bagaimana data dapat dikembalikan ke kondisi awal secara real-time.

Secara keseluruhan, praktikum ini membantu memahami perbedaan antara StatelessWidget dan StatefulWidget, sekaligus menunjukkan bagaimana Flutter dapat digunakan untuk membuat aplikasi yang interaktif, responsif, dan menarik secara visual.