LAPORAN PRAKTIKUM MOBILE PROGRAMMING
PERTEMUAN 11

Nama : Faril Isra Albiansyah

NIM : 240605110087

Kelas : Mobile Programming B

Tanggal : 13 – 10 – 2025

JURUSAN TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM

MALANG

GANJIL 2025/2026

I. Tujuan

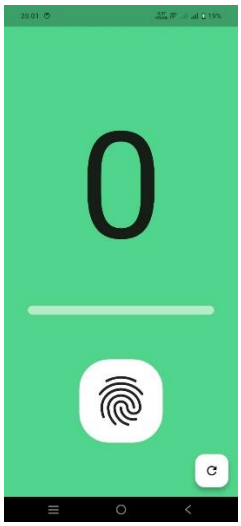
Memahami konsep manajemen state menggunakan GetX, menerapkan controller dan variabel reaktif dalam pembaruan tampilan, serta membandingkan penggunaan GetX dengan pendekatan konvensional setState() pada Flutter.

III. Langkah Kerja

- Membuat proyek Flutter baru dengan nama getx_app.
- Menambahkan dependensi pada file pubspec.yaml.
- Membuat struktur folder yang terdiri atas model, view, dan viewmodel.
- Membuat kelas TasbihController yang berisi variabel reaktif counter dan progress, serta method incrementCounter() dan resetCounter().
- Menghubungkan controller dengan tampilan menggunakan widget Obx.
- Menjalankan aplikasi untuk melihat perubahan nilai counter dan progress bar secara realtime.

IV. Screenshot Hasil

1. Tampilan Awal Aplikasi

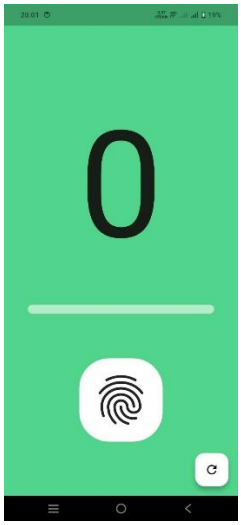


Keterangan: Screenshot ini menampilkan kondisi awal aplikasi saat pertama kali dijalankan. Nilai penghitung masih 0, dan progress bar dalam keadaan kosong.

2. Tampilan Saat Penghitungan Berlangsung

Keterangan: Screenshot ini diambil setelah tombol fingerprint ditekan beberapa kali (contohnya 17 kali). Angka pada penghitung berubah menjadi 17, dan progress bar terisi sebagian, menunjukkan progres hitungan menuju 33. Perubahan ini terjadi secara otomatis tanpa setState().

3. Tampilan Setelah direset



Keterangan: Screenshot ini menunjukkan kondisi aplikasi setelah FloatingActionButton (tombol reset) ditekan. Nilai penghitung dan progress bar langsung kembali ke 0, membuktikan bahwa method resetCounter() berfungsi dengan baik.

Pembahasan: Perbedaan setState() dan GetX

Dalam praktikum ini, perbedaan mendasar antara pendekatan setState() dan GetX menjadi sangat jelas.

setState() (Konvensional): Ketika setState() dipanggil, Flutter akan memberi tahu framework untuk menjadwalkan pembangunan ulang (rebuild) untuk seluruh widget tree dari widget tersebut. Pada aplikasi kompleks, hal ini bisa menjadi tidak efisien karena banyak widget yang tidak berubah ikut di-rebuild, yang dapat memengaruhi performa.

GetX (Reaktif): GetX menggunakan pendekatan yang lebih efisien. Dengan membungkus widget yang perlu berubah menggunakan Obx, hanya widget tersebut yang akan di-rebuild ketika nilai variabel reaktif (.obs) yang dipantaunya berubah. Ini berarti pembaruan UI lebih terisolasi dan ringan, sehingga performa aplikasi menjadi lebih baik, terutama pada skala yang lebih besar. Selain itu, GetX memisahkan logika bisnis (di dalam Controller) dari tampilan (di dalam View), membuat kode lebih bersih dan mudah dikelola.

IV. Kesimpulan

Berdasarkan praktikum yang telah dilaksanakan, dapat ditarik beberapa kesimpulan:

- Efisiensi Pembaruan UI: GetX memungkinkan pembaruan UI secara otomatis dan efisien hanya pada widget yang diperlukan melalui Obx dan variabel reaktif (.obs), tanpa perlu memanggil setState() yang me-rebuild seluruh widget tree.
- Struktur Kode yang Terorganisir: Penggunaan Controller pada GetX berhasil memisahkan antara logika bisnis aplikasi dan kode tampilan (UI). Hal ini menjadikan arsitektur aplikasi lebih bersih, terstruktur, dan mudah untuk dipelihara di kemudian hari.
- Pengembangan Responsif: Dengan menghubungkan aksi pengguna langsung ke method di dalam controller, aplikasi menjadi sangat interaktif

dan responsif, di mana setiap perubahan state langsung tercermin pada tampilan secara real-time.