LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI BERGERAK

Android Fundamental 5 - WEEK 11

Disusun oleh:

Nama : Alfath Roziq Widhayaka

Nim : L0122012

Kelas : A

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

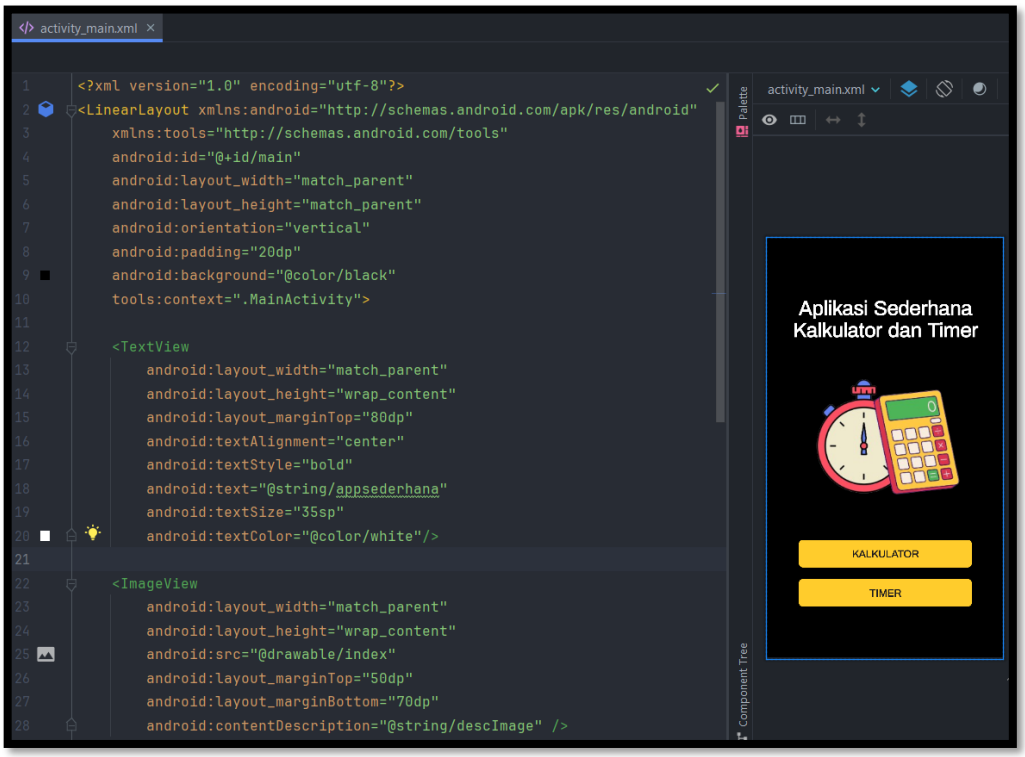
2024

1. **Screenshot Source Code**

Pada praktikum Pengembangan Aplikasi Bergerak pada minggu ke-11 yang saya buat ini merupakan aplikasi sederhana dengan fitur **Kalkulator** dan **Timer**. Pembuatan aplikasi ini menerapkan materi yang diberikan, yaitu **ViewModel** dan **LiveData**. Penggunaan ViewModel disini digunakan untuk menyimpan dan mengelola data terkait dengan antarmuka user (UI) dalam kalkulator dan timer ini, sedangkan LiveData digunakan untuk menyediakan mekanisme dari data waktu yang sedang berjalan pada timer. Berikut merupakan source code pada aplikasi yang saya buat.

A. Membuat Halaman Awal

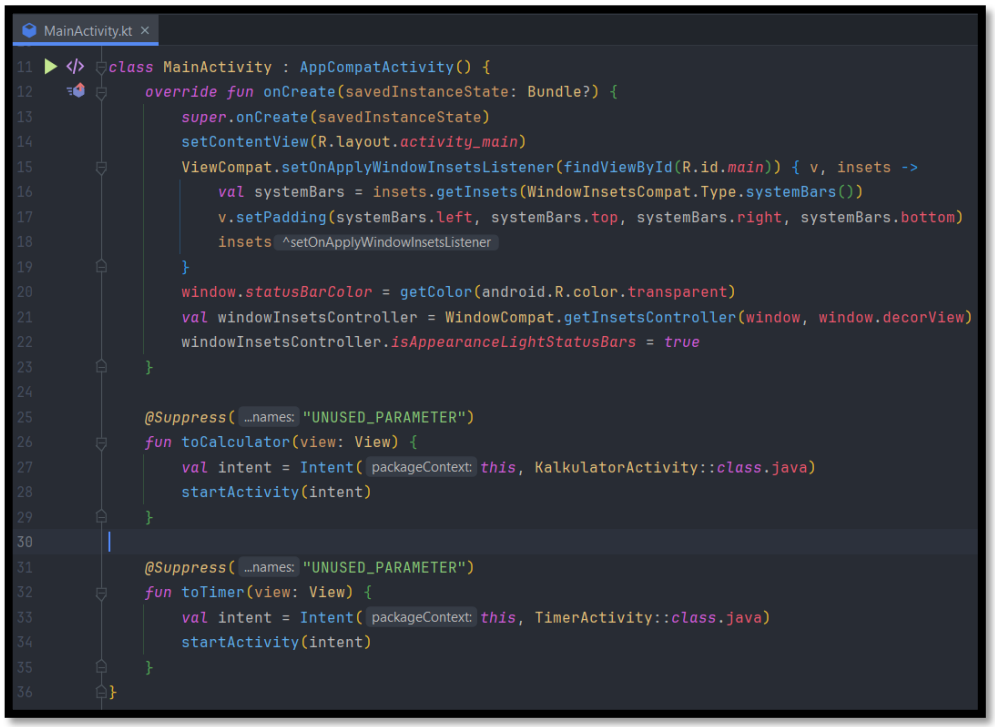
- **activity_main.xml**



Gambar 1 activity_main.xml

Kode XML milik MainActivity diatas merupakan tata letak antarmuka user untuk aplikasi Android. Terdapat **LinearLayout** yang menyusun elemen-elemennya secara vertikal dengan latar belakang hitam dan padding 20dp. Didalamnya, terdapat judul dengan **TextView**, gambar **ImageView**, dan dua tombol **AppCompatButton**. Setiap elemen memiliki atribut-atribut seperti ukuran, gaya teks, dan tampilan yang telah dikustomisasi. Tombol-tombol tersebut memiliki atribut onClick yang menghubungkannya dengan metode tertentu atau halaman selanjutnya yang dipilih untuk menangani aksi user.

- **MainActivity.kt**

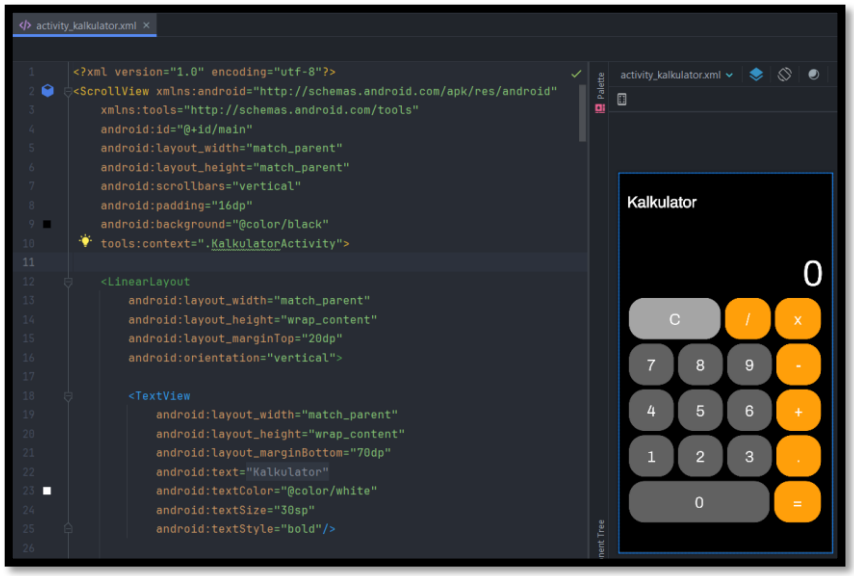


Gambar 2 MainActivity.kt

Kode MainActivity.kt diatas digunakan untuk mengatur fungsi aplikasi Android. Metode onCreate mengaitkan layout XML dengan kelas MainActivity, menyesuaikan padding dengan area sistem, dan mengatur warna status bar menjadi transparan. Terdapat dua metode yang digunakan disini, yaitu **toCalculator** dan **toTimer**, mengarahkan user ke aktivitas **Kalkulator** dan **Timer** saat masing-masing tombol di klik.

B. Membuat Halaman Kalkulator

- **activity_kalkulator.xml**



Gambar 3 activity_kalkulator.xml

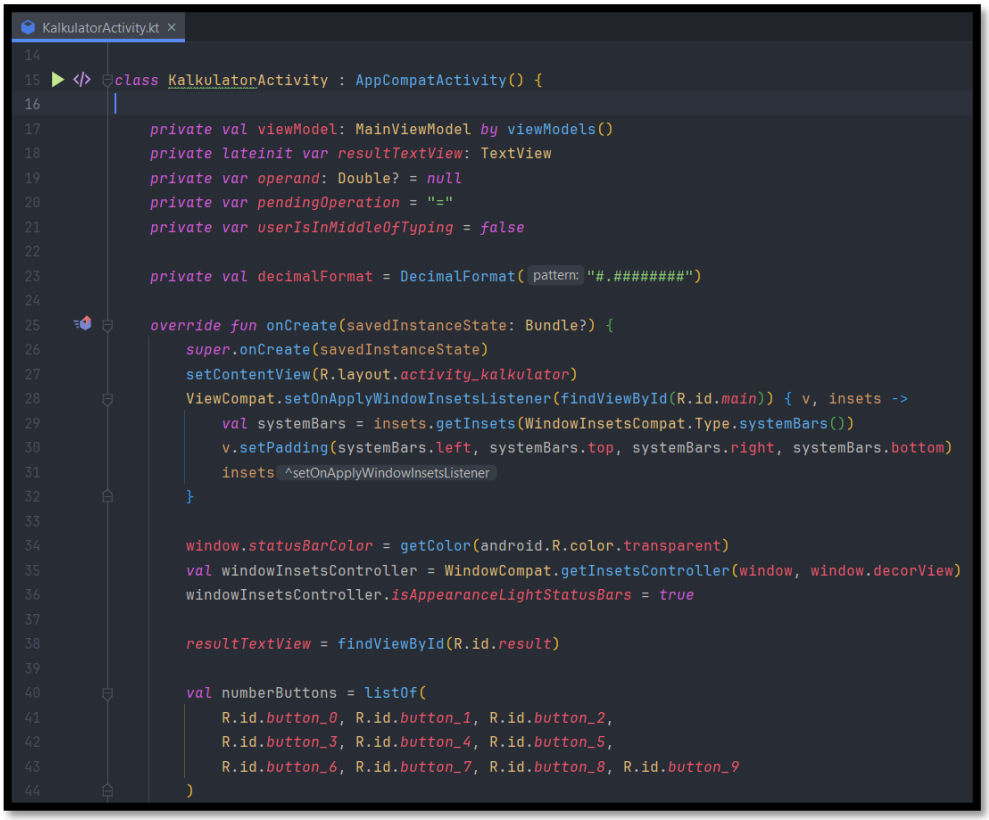
Kode XML milik acticity_kalkulaor.xml di atas adalah tata letak antarmuka user untuk sebuah kalkulator Android. Ini terdiri dari **ScrollView** yang memungkinkan user untuk menggulir konten, dengan **LinearLayout** sebagai wadah utama. Di dalamnya, terdapat **TextView** untuk menampilkan hasil dan sejumlah tombol untuk input angka dan operasi matematika. Setiap tombol memiliki atribut seperti identifikasi, ukuran, teks, dan latar belakang yang telah ditentukan. Semua ini menggambarkan struktur dasar dari tata letak kalkulator yang memungkinkan user untuk memasukkan angka, melakukan operasi matematika, dan melihat hasilnya.

- **MainViewModel.kt**

Gambar 4 MainViewModel.kt Kalkulator

Kode MainViewmModel.kt di atas adalah bagian dari aplikasi Android yang mengimplementasikan materi **ViewModel**. Bagian ViewModel milik kalkulator disini bertanggung jawab untuk logika perhitungan dan menyediakan data yang diperlukan oleh antarmuka user (UI). Dalam konteks kalkulator, **ViewModel** memiliki fungsi-fungsi yang digunakan seperti **add**, **subtract**, **multiply**, dan **divide** untuk melakukan operasi matematika dasar. Pendekatan ini memisahkan logika perhitungan dari antarmuka user, sehingga memungkinkan pengujian yang lebih mudah dan pemisahan tugas yang bersih.

- **KalkulatorActivity.kt**



KalkulatorActivity.kt

```kotlin
    val operationButtons = listOf(
        R.id.button_add, R.id.button_subtract, R.id.button_multiply, R.id.button_divide, R.id.button_equals
    )

    val clearButton: Button = findViewById(R.id.button_clear)
    clearButton.setOnClickListener { it: View!
        operand = null
        pendingOperation = "="
        resultTextView.text = "0"
        userIsInMiddleOfTyping = false
    }

    val dotButton: Button = findViewById(R.id.button_dot)
    dotButton.setOnClickListener { it: View!
        val resultText = resultTextView.text.toString()
        if (!resultText.contains( other: ".")) {
            resultTextView.append(".")
            userIsInMiddleOfTyping = true
        }
    }

    val listener = View.OnClickListener { v ->
        val button = v as Button
        val text = button.text.toString()
        if (userIsInMiddleOfTyping) {
            resultTextView.append(text)
        } else {
            resultTextView.text = text
            userIsInMiddleOfTyping = true
        }
    }
```

KalkulatorActivity.kt

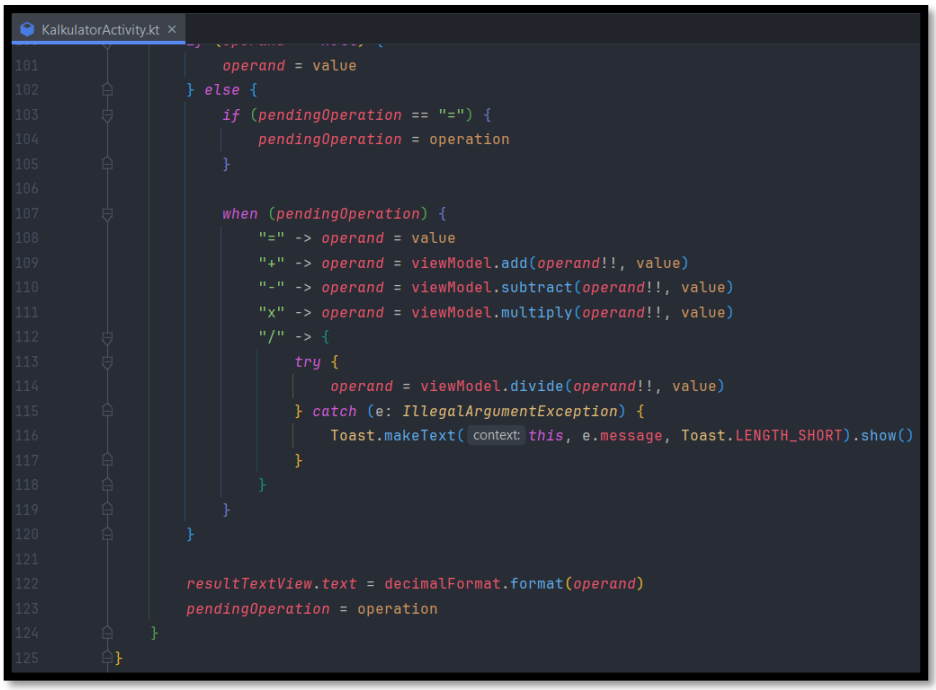
```kotlin
        for (buttonId in numberButtons) {
            findViewById<Button>(buttonId).setOnClickListener(listener)
        }

        val opListener = View.OnClickListener { v ->
            try {
                val value = resultTextView.text.toString().toDouble()
                val button = v as Button
                val operation = button.text.toString()
                performOperation(value, operation)
            } catch (e: NumberFormatException) {
                resultTextView.text = ""
            }
            userIsInMiddleOfTyping = false
        }

        for (buttonId in operationButtons) {
            findViewById<Button>(buttonId).setOnClickListener(opListener)
        }
    }

    private fun performOperation(value: Double, operation: String) {
        if (operand == null) {
            operand = value
        } else {
            if (pendingOperation == "=") {
                pendingOperation = operation
            }

            when (pendingOperation) {
                "=" -> operand = value
```

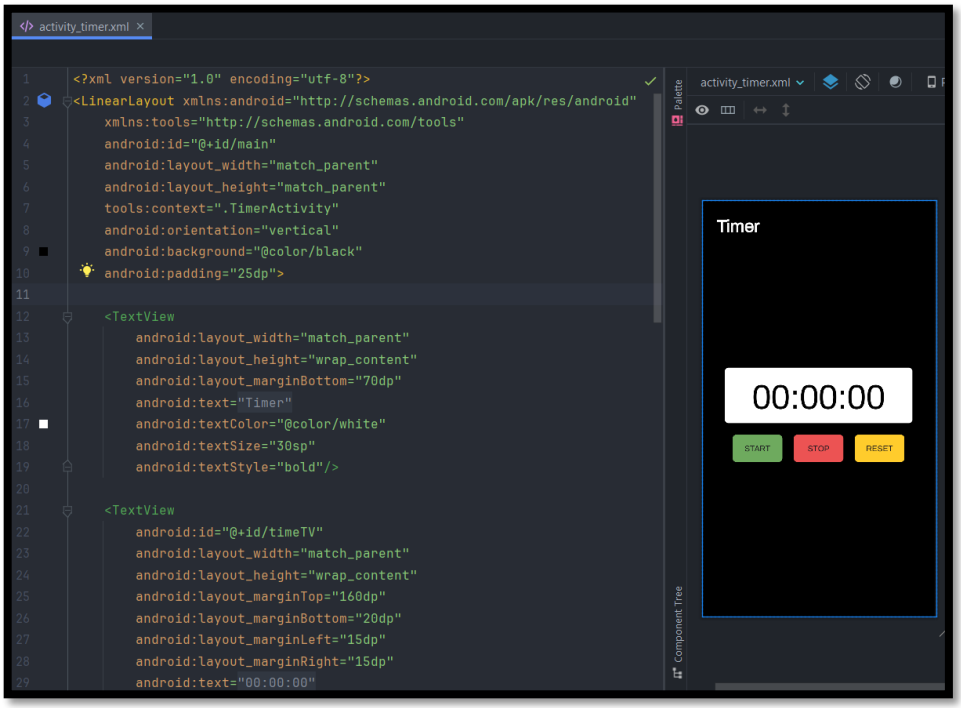


Gambar 5 KalkulatorActivity.kt

Kode KalkulatorActivity.kt di atas merupakan implementasi dari sebuah kalkulator sederhana dalam aplikasi Android. Aktivitas ini mengikat **ViewModel** dari file **MainViewModel.kt** untuk mengelola masing-masing operasi matematika. Disini terdapar juga pengaturan untuk tombol-tombol **angka**, **operasi**, **clear**, dan **koma**. Ketika tombol-tombol ini diklik, isi dari tombol tersebut mengirimkan input ke layar kalkulator. Metode **performOperation()** melakukan operasi matematika berdasarkan operasi yang dipilih oleh user. Jika terjadi kesalahan dalam mengonversi teks menjadi angka, exception akan ditangani dan tindakan yang sesuai akan diambil. Hasil dari operasi dihitung dan ditampilkan pada **resultTextView** dengan format yang ditentukan. Dengan menggunakan **ViewModel** hal ini dapat memisahkan tampilan dari logika perhitungan pada aplikasi, sehingga memudahkan pengujian/eksekusi dan pengelolaan kode secara terpisah.

C. Membuat Halaman Timer

- **activity_timer.xml**



Gambar 6 activity_timer.xml

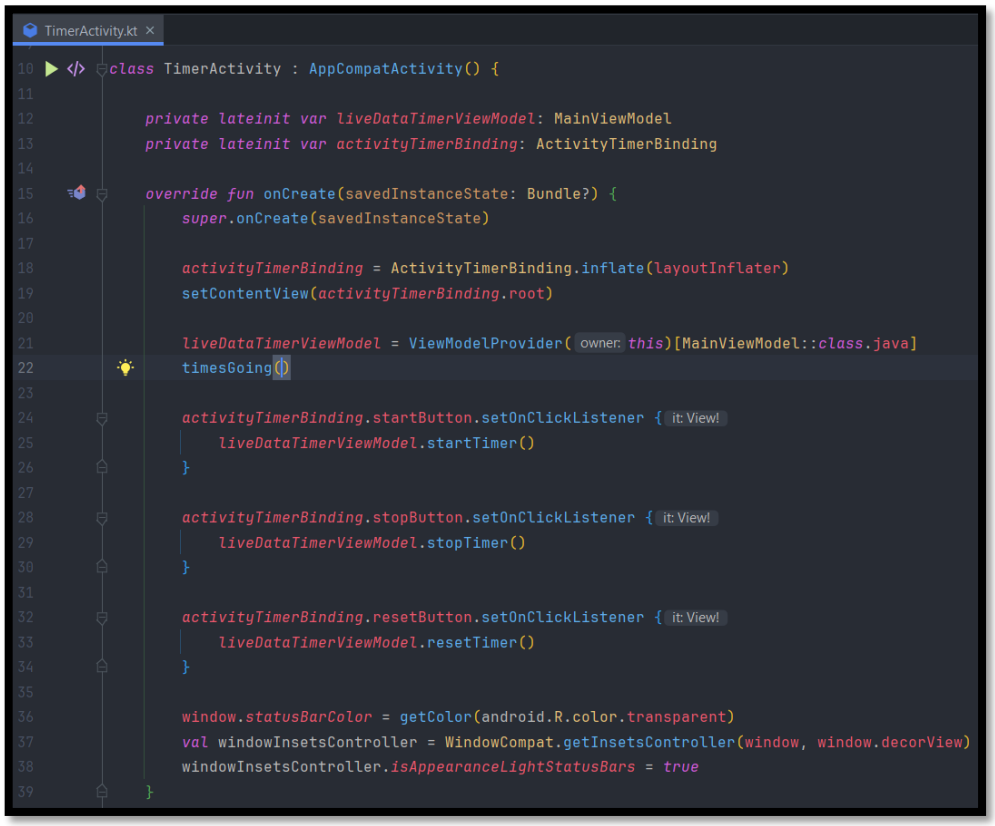
Kode XML di atas mendefinisikan tata letak untuk halaman Timer dalam aplikasi Android menggunakan **LinearLayout** sebagai wadah utama. Halaman ini memiliki latar belakang hitam dengan padding 25dp di sekelilingnya. Terdapat dua **TextView** yang menampilkan teks, di mana yang pertama memiliki ukuran teks besar dengan warna putih dan teks **Timer** yang ditebalkan, sedangkan yang kedua menampilkan waktu dengan ukuran besar dan teks warna hitam di tengah layar, dikelilingi oleh kotak dengan latar belakang tertentu. Di bawahnya, terdapat tiga tombol (**start, stop, reset**) yang dikelompokkan dalam **LinearLayout** horizontal dengan **gravity center**. Setiap tombol memiliki fungsi tertentu dan ditampilkan dengan teks yang sesuai.

- **MainViewModel.kt**

Gambar 7 MainViewModel.kt Timer

Kode MainViewModel.kt diatas merupakan fungsi-fungsi yang digunakan pada halaman Timer ini. **ViewModel** digunakan untuk menyimpan data yang berhubungan dengan waktu yang berlalu. **LiveData** digunakan untuk mengamati perubahan waktu tersebut dan memberitahu tampilan ketika ada perubahan. Ketika timer dimulai, waktu mulai disimpan, dan **Runnable** dijalankan untuk menghitung waktu yang telah berlalu setiap 1 detik. Selanjutnya, **LiveData** yang menyimpan waktu yang telah berlalu diperbarui dan diamati oleh antarmuka user untuk ditampilkan. Saat timer dihentikan atau diatur ulang, **Runnable** dihapus untuk menghentikan perhitungan waktu. Dengan menggunakan **ViewModel** dan **LiveData**, aplikasi dapat menjaga konsistensi data selama siklus hidup aktivitas Android dan menghindari kehilangan data saat rotasi layar atau perubahan konfigurasi lainnya.

- **TimerActivity.kt**



Gambar 8 TimerActivity.kt

Kode **TimerActivity.kt** di atas merupakan bagian dari sebuah aplikasi Android yang mengimplementasikan **ViewModel** dan **LiveData** untuk mengatur dan menampilkan waktu yang telah berlalu pada suatu timer. Saat aktivitas TimerActivity dibuat, **ViewModelProvider** digunakan untuk mendapatkan instance dari **MainViewModel.kt**. Fungsi **timesGoing()** menghubungkan Observer ke **LiveData** yang menyimpan waktu yang telah berlalu, sehingga setiap kali waktu berubah, antarmuka user akan diperbarui. Metode **onCreate()** menetapkan fungsi-fungsi **onClickListener** untuk tombol **mulai, stop, dan reset**. Dengan menggunakan ViewModel dan LiveData, aplikasi dapat memisahkan logika perhitungan dari antarmuka user dan mengamati perubahan data secara reaktif untuk memperbarui UI.

2. **Screenshot Terminal**

Berikut adalah hasil pada emulator ketika aplikasi dijalankan atau dieksekusi. Aplikasi ini menerapkan materi **ViewModel** dan **LiveData** untuk menyimpan dan mengelola data terkait dengan antarmuka user (UI) dalam kalkulator dan timer ini, serta menyediakan mekanisme dari data waktu yang sedang berjalan pada timer. Berikut merupakan hasil screenshoot pada aplikasi yang saya buat.

A. Halaman Awal

Gambar 9 Halaman Awal

Gambar diatas merupakan hasil dari emulator saat aplikasi dijalankan. Tampilan ini merupakan halaman awal yang menampilkan judul aplikasi, gambar sederhana, dan dua tombol yang dapat mengarah ke masing-masing halaman yang user ingin buka. Jika user memilih tombol kalkulator maka akan menuju ke halaman **kalkulator**, begitu juga dengan tombol timer maka akan menuju ke halaman **timer**. Tampilan antarmuka diatas disusun sedemikian rupa untuk memudahkan user dalam menggunakan aplikasi.

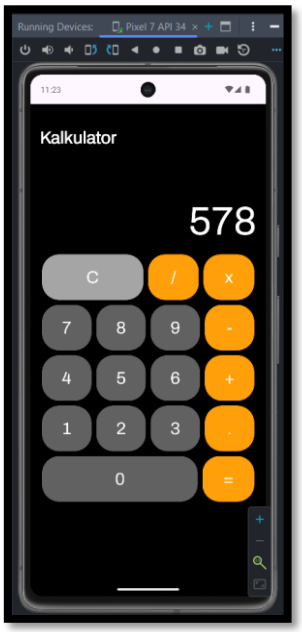
B. Halaman Kalkulator

- **Halaman Awal Kalkulator**

Gambar 10 Halaman Kalkulator Awal

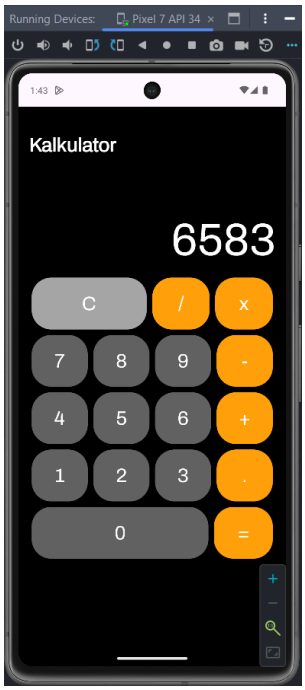
Gambar diatas merupakan tampilan ketika user telah menekan tombol Kalkulator dan masuk ke halaman **Kalkulator**. Pada halaman ini menampilkan kalkulator sederhana yang tampilannya sama atau mirip dengan hp pada umumnya. Terdapat banyak tombol yang berisi angka, macam operasi, koma, dan clear. Masing-masing tombol dapat digunakan dengan baik oleh user sehingga meningkatkan efisiensi aplikasi.

- **Operasi penjumlahan (78 + 58)**



Gambar 11 Halaman Kalkulator penjumlahan

- **Operasi pengurangan (7546 – 963)**



Gambar 12 Halaman Kalkulator pengurangan

- **Operasi perkalian (144 x 78)**

Gambar 13 Halaman Kalkulator perkalian

- **Operasi pembagian (4563 : 96)**

Gambar 14 Halaman Kalkulator pembagian

- **Operasi koma (785.63 x 6)**

Gambar 15 Halaman Kalkulator perkalian koma

C. Halaman Timer

Gambar 16 Halaman Timer

Gambar diatas merupakan tampilan ketika user telah menekan tombol Timer lalu masuk ke halaman **Timer**. Pada halaman ini menampilkan Timer sederhana yang dapat digunakan oleh user. Fungsi **start, stop,** dan **reset** semuanya dapat berfungsi dengan baik ketika dijalankan. Tampilan antarmuka dibuat semenarik dan se sederhana mungkin dikarenakan yang dibutuhkan hanya kegunaannya saja.

3. Kesimpulan

Pada praktikum Pengembangan Aplikasi Bergerak minggu ke-11 ini, saya berhasil mengembangkan aplikasi Android dengan fitur **Kalkulator** dan **Timer** menggunakan konsep **ViewModel** dan **LiveData**. **ViewModel** digunakan untuk mengelola data dan logika perhitungan secara efisien. Sementara itu, **LiveData** digunakan untuk pengamatan terhadap perubahan data, seperti waktu yang berjalan pada fitur **Timer**. Implementasi ini memastikan aplikasi tetap responsif dan konsisten meskipun terjadi perubahan konfigurasi seperti rotasi layar.

Penggunaan **ViewModel** dalam Kalkulator memudahkan pengelolaan operasi matematika, sedangkan **LiveData** pada Timer memastikan pembaruan waktu secara dinamis. Implementasi **ViewModel** dan **LiveData** meningkatkan pengalaman user dengan menyediakan tampilan yang konsisten dan data yang akurat.