LAPORAN PRAKTIKUM

PENGEMBANGAN APLIKASI BERGERAK

PRAKTIKUM 11 – VIEW MODEL DAN LIVE DATA

L0122034

BINTANG HARIDA RAMADHAN

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

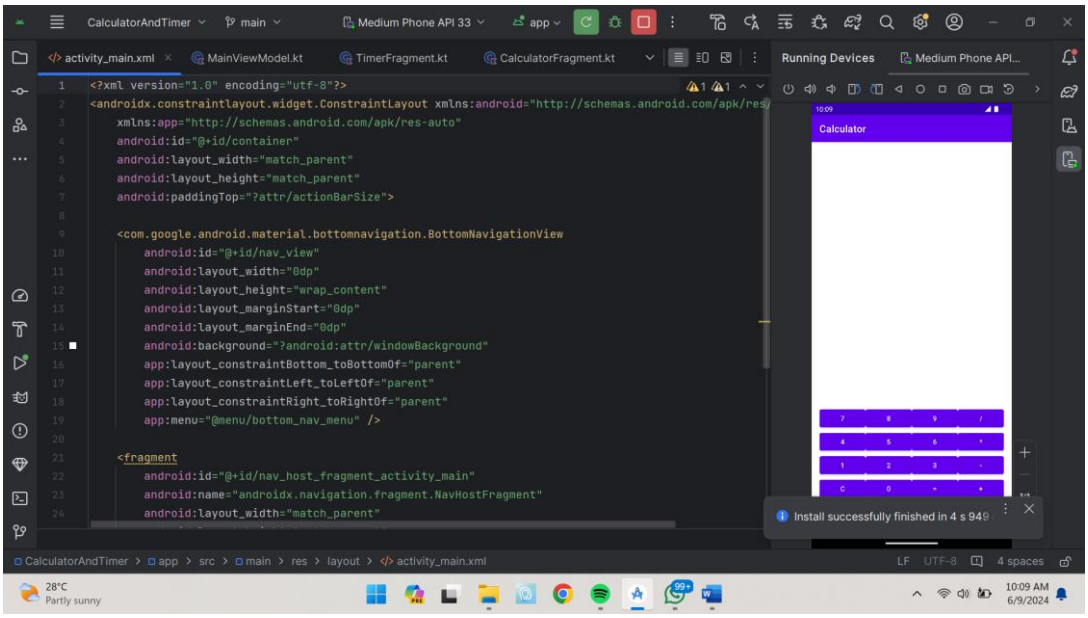
UNIVERSITAS SEBELAS MARET

2024

BAB I

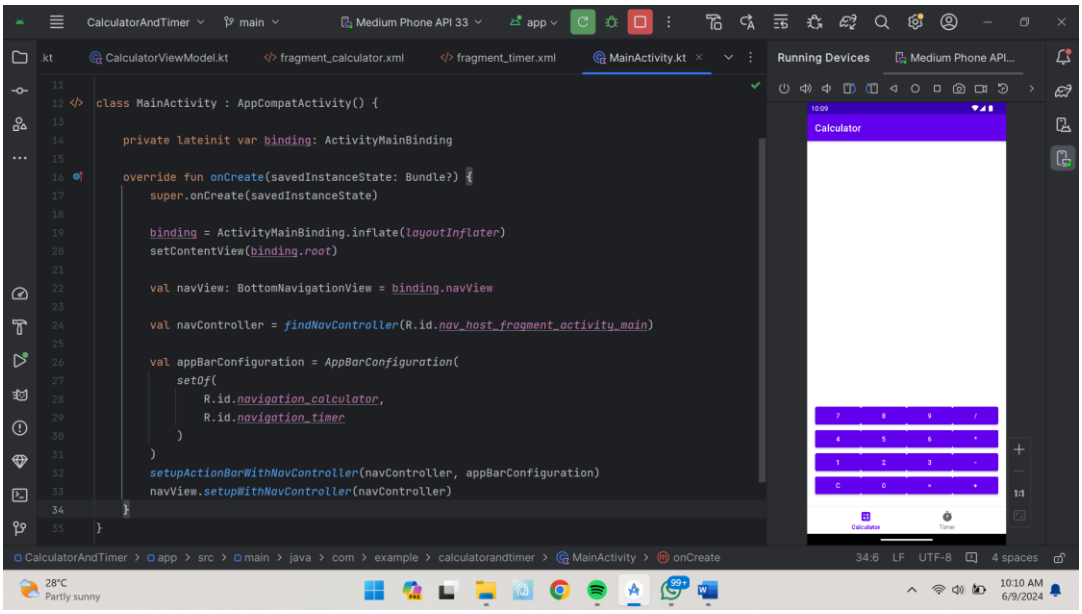
SOURCE CODE

1. File “activity_main.xml”



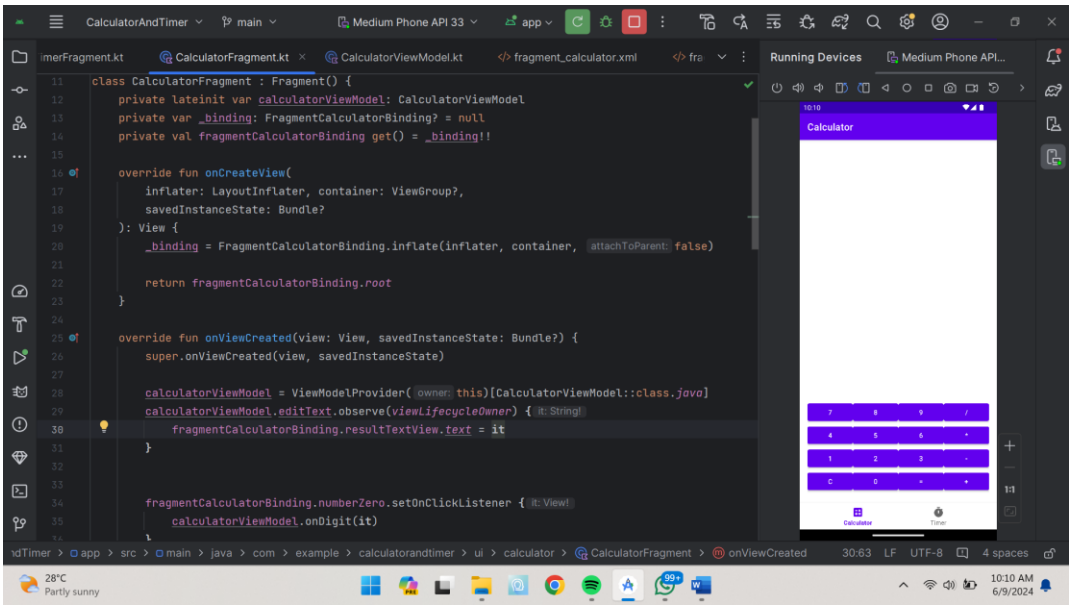
Source code di atas mendefinisikan layout untuk aktivitas utama dalam aplikasi Android yang menggunakan ConstraintLayout sebagai root view. Layout ini berisi dua komponen utama: BottomNavigationView dan NavHostFragment. BottomNavigationView (dengan id nav_view) ditempatkan di bagian bawah layar dan menyediakan navigasi melalui menu yang ditentukan di @menu/bottom_nav_menu. NavHostFragment (dengan id nav_host_fragment_activity_main) mengambil sisa ruang layar dan berfungsi sebagai container untuk fragment navigasi yang dikelola oleh NavController, dengan graf navigasi yang ditentukan di @navigation/mobile_navigation. Komponen ini diatur menggunakan constraint untuk memastikan mereka saling berhubungan dengan benar dalam layout.

2. File “MainActivity.kt”



MainActivity adalah aktifitas utama dalam aplikasi Android yang menggunakan navigasi berbasis fragment dengan BottomNavigationView. Aktivitas ini menggunakan data binding untuk mengikat layout dengan ActivityMainBinding. Pada metode onCreate, binding diinisialisasi dan root view diatur sebagai konten tampilan. BottomNavigationView diambil dari binding dan digunakan untuk mengatur navigasi. NavController ditemukan menggunakan ID fragment host navigasi (R.id.nav_host_fragment_activity_main). Konfigurasi AppBar dibuat untuk mengelola destinasi navigasi utama (R.id.navigation_calculator dan R.id.navigation_timer). Fungsi setupActionBarWithNavController dan setupWithNavController digunakan untuk menghubungkan NavController dengan ActionBar dan BottomNavigationView, memungkinkan navigasi antara kalkulator dan timer dengan menggunakan BottomNavigationView.

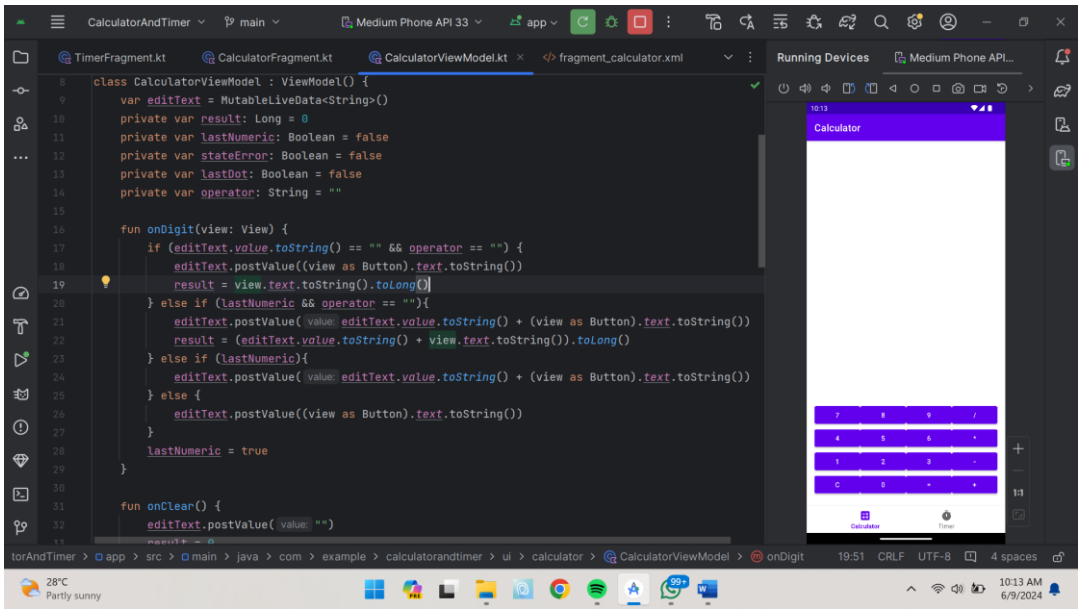
3. File “CalculatorFragment.kt”



```
class CalculatorFragment : Fragment() {
    private lateinit var calculatorViewModel: CalculatorViewModel
    private var _binding: FragmentCalculatorBinding? = null
    private val fragmentCalculatorBinding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentCalculatorBinding.inflate(inflater, container, attachToParent: false)

        return fragmentCalculatorBinding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        calculatorViewModel = ViewModelProvider( owner: this)[CalculatorViewModel::class.java]
        calculatorViewModel.editText.observe(viewLifecycleOwner) { it: String!
            fragmentCalculatorBinding.resultTextView.text = it
        }


        fragmentCalculatorBinding.numberZero.setOnClickListener { it: View!
            calculatorViewModel.onDigit(it)
```

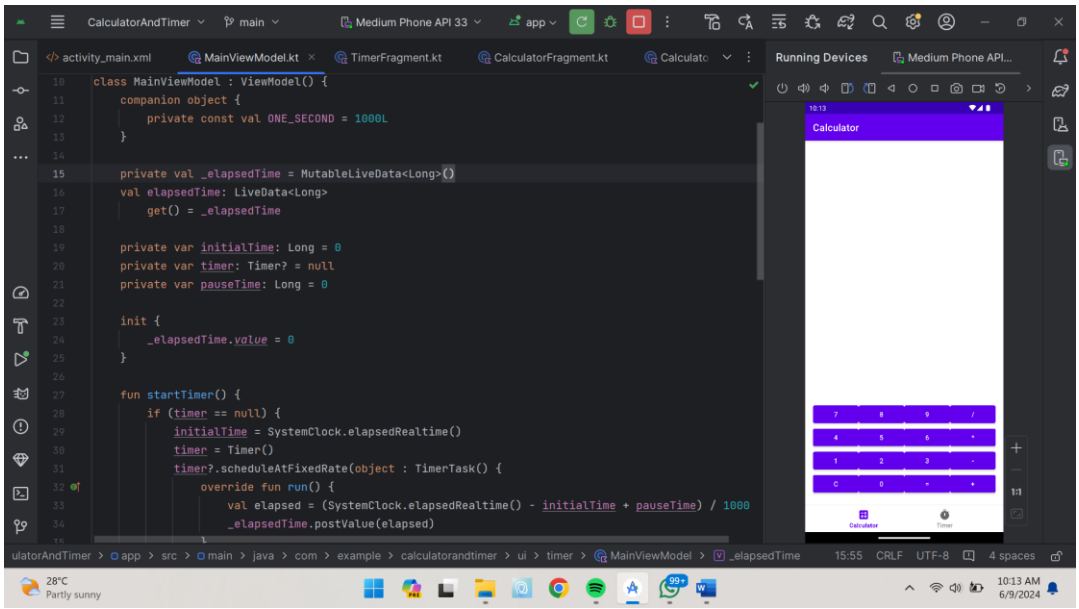
Source Code di atas merupakan implementasi dari CalculatorFragment, yaitu sebuah fragment dalam aplikasi Android yang menampilkan kalkulator dan mengelola interaksi pengguna menggunakan ViewModel. Fragment ini menggunakan data binding dengan FragmentCalculatorBinding untuk mengikat tampilan dengan kode. Dalam onCreateView, binding diinisialisasi dengan FragmentCalculatorBinding.inflate. ViewModelProvider digunakan untuk mendapatkan instance CalculatorViewModel, yang mengelola logika kalkulator. LiveData dalam CalculatorViewModel dipantau, dan hasilnya ditampilkan di resultTextView. Berbagai tombol pada kalkulator diatur dengan setOnClickListener untuk memanggil metode yang sesuai di CalculatorViewModel, seperti onDigit, onOperator, onEqual, dan onClear, untuk memperbarui tampilan dan memproses input pengguna. Binding dibersihkan saat fragment dihancurkan untuk mencegah kebocoran memori.

4. File “CalculatorViewModel.kt”



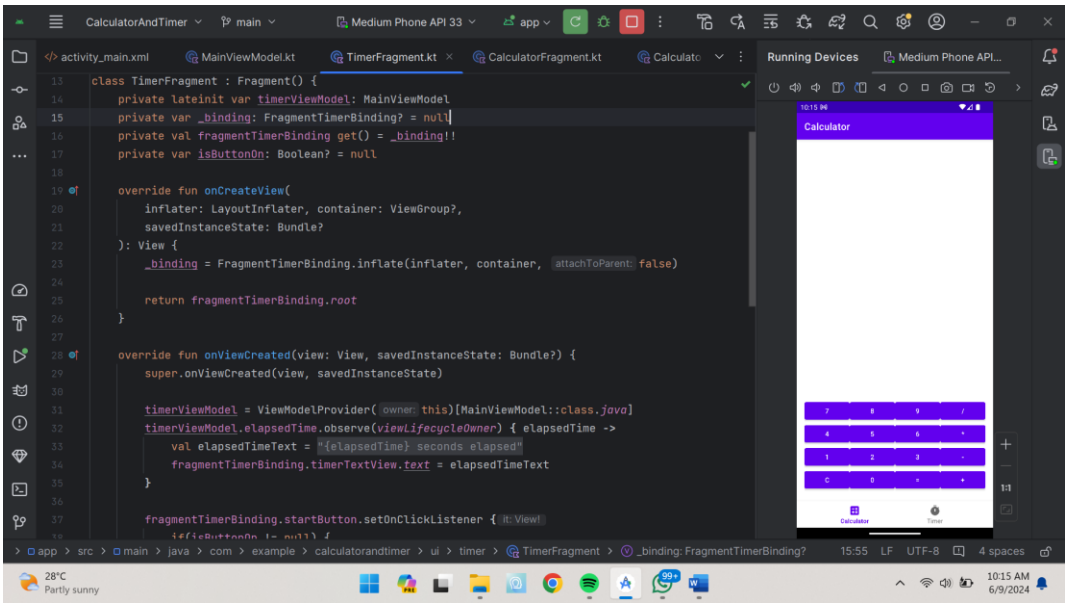
Source Code di atas mendefinisikan sebuah ViewModel untuk mengelola logika bisnis dari aplikasi kalkulator berbasis MVVM (Model-View-ViewModel) di Android. CalculatorViewModel menggunakan MutableLiveData untuk memperbarui tampilan pengguna secara reaktif. Terdapat dua variabel LiveData: editText untuk menyimpan dan memperbarui nilai yang ditampilkan di layar kalkulator, dan liveResult untuk menyimpan hasil perhitungan. Variabel internal seperti result, lastNumeric, stateError, lastDot, dan operator digunakan untuk melacak status dan operasi kalkulator saat ini. Metode onDigit menambahkan digit ke tampilan, memperbarui editText, dan mengatur flag lastNumeric. Metode onClear menghapus semua nilai dan mereset status kalkulator. Metode onEqual menghitung hasil berdasarkan operator yang digunakan terakhir dan memperbarui editText dengan hasilnya. Metode onOperator mengatur operator matematika yang dipilih dan memperbarui tampilan. ViewModel ini memungkinkan kalkulator untuk memproses input pengguna dan menampilkan hasil secara reaktif di UI.

5. File “MainViewModel.kt”



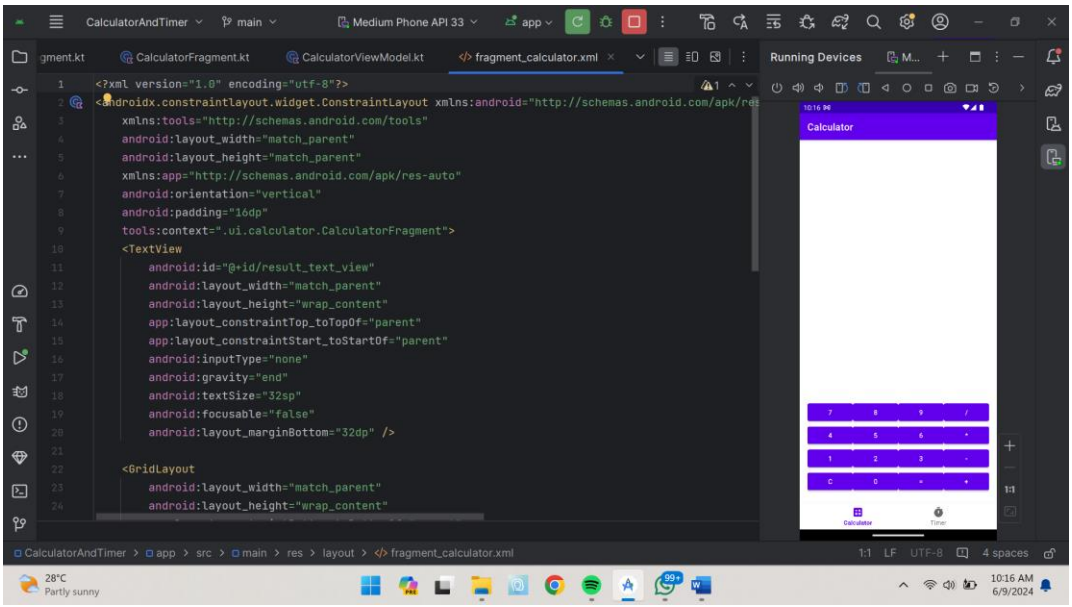
Source Code di atas mendefinisikan sebuah ViewModel untuk mengelola logika dari fitur timer dalam aplikasi Android berbasis MVVM (Model-View-ViewModel). ViewModel ini menggunakan MutableLiveData untuk memperbarui tampilan pengguna secara reaktif melalui LiveData. Variabel _elapsedTime menyimpan waktu yang telah berlalu dalam detik, dan elapsedTime memberikan akses read-only ke variabel ini. ViewModel ini memiliki fungsi untuk mengelola timer, seperti startTimer, pauseTimer, resumeTimer, dan stopTimer. Fungsi startTimer menginisialisasi dan memulai timer yang menghitung waktu yang telah berlalu. Fungsi pauseTimer menghentikan timer sementara dan menyimpan waktu yang telah berlalu. Fungsi resumeTimer melanjutkan timer dari waktu yang telah dihentikan. Fungsi stopTimer menghentikan timer dan mengatur ulang waktu yang telah berlalu. Metode onCleared memastikan bahwa timer dibatalkan ketika ViewModel dihancurkan, untuk mencegah kebocoran memori.

6. File “TimerFragment.kt”



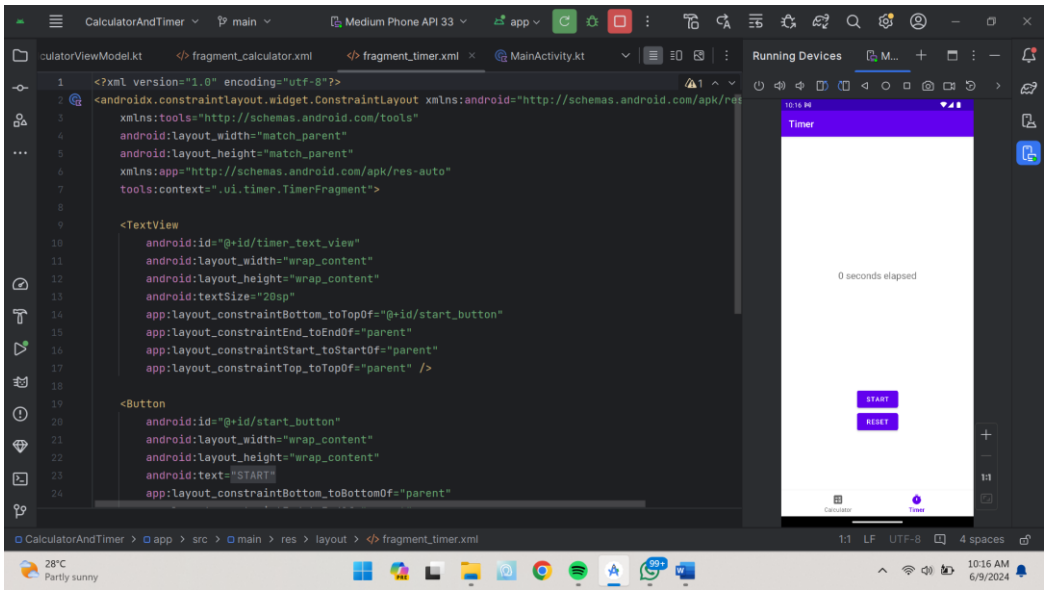
Source Code mendefinisikan sebuah fragment yang menampilkan dan mengelola fungsi timer dalam aplikasi Android. Fragment ini menggunakan data binding dengan FragmentTimerBinding untuk mengikat tampilan dengan kode. Dalam onCreateView, binding diinisialisasi dengan FragmentTimerBinding.inflate. ViewModelProvider digunakan untuk mendapatkan instance MainViewModel, yang mengelola logika timer. LiveData dalam MainViewModel dipantau, dan waktu yang telah berlalu diperbarui di timerTextView. Tombol startButton diatur untuk memulai, menjeda, atau melanjutkan timer berdasarkan status tombol (isButtonOn). Jika timer dimulai, teks tombol berubah menjadi "Pause Timer", jika dijeda, teks tombol berubah menjadi "Resume Timer". Tombol resetButton diatur untuk mengatur ulang timer dan mengubah teks tombol startButton menjadi "Start Timer". Binding dibersihkan dalam onDestroyView untuk mencegah kebocoran memori.

7. File “fragment_calculator.xml”



Source code di atas merupakan layout untuk CalculatorFragment dalam aplikasi Android. Layout ini menggunakan ConstraintLayout sebagai root view dan mengatur tata letak vertikal. Terdapat sebuah TextView untuk menampilkan hasil perhitungan kalkulator, diikuti dengan sebuah GridLayout yang berisi tombol-tombol kalkulator. Setiap tombol kalkulator (angka, operasi matematika, dan tombol Clear) diberi bobot kolom yang sama agar memiliki lebar yang sama. GridLayout ini diatur untuk memiliki empat baris dan empat kolom, dengan jarak bawah yang cukup untuk memisahkan tombol-tombol dari bagian bawah layar. Semua elemen UI diatur menggunakan constraint untuk memastikan tampilan yang sesuai dalam layout keseluruhan.

8. File “fragment_timer.xml”

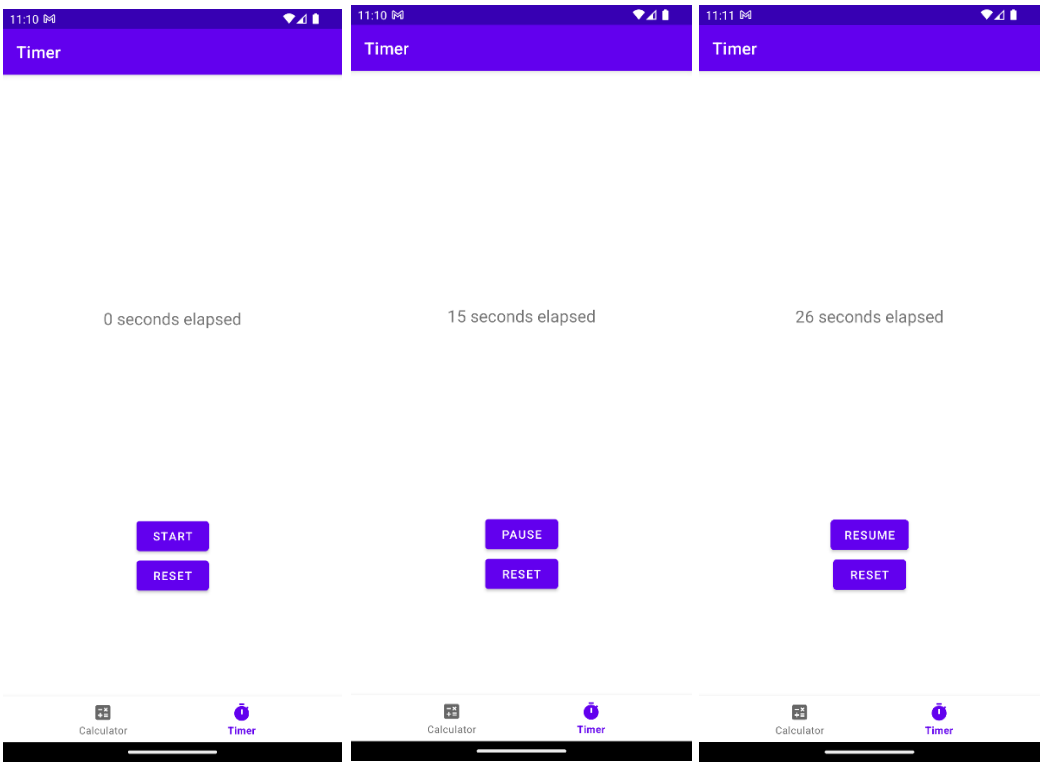


Source code di atas merupakan layout untuk TimerFragment dalam aplikasi Android. Layout ini menggunakan ConstraintLayout sebagai root view dan mengatur tata letak vertikal. Terdapat sebuah TextView untuk menampilkan waktu yang telah berlalu dalam timer. Di bawahnya, terdapat dua tombol, yaitu tombol start_button untuk memulai dan menjeda timer, serta tombol reset_button untuk mengatur ulang timer. Semua elemen UI diatur menggunakan constraint untuk memastikan tampilan yang sesuai dalam layout keseluruhan.

BAB II
USER INTERFACE

1. Calculator Fragment

2. Timer Fragment



BAB III
KESIMPULAN

ViewModel dan LiveData adalah komponen inti dalam arsitektur MVVM (Model-View-ViewModel) di Android Studio yang membantu dalam memisahkan logika bisnis dari antarmuka pengguna serta mengelola data secara efisien. ViewModel dirancang untuk menyimpan dan mengelola data UI agar tetap konsisten meskipun terjadi perubahan konfigurasi seperti rotasi layar. LiveData, di sisi lain, adalah kelas pembungkus yang memungkinkan data yang dapat diamati dan bereaksi terhadap perubahan data, sehingga UI dapat diperbarui secara otomatis ketika data berubah. Kombinasi kedua komponen ini memungkinkan pengembangan aplikasi yang lebih modular, mudah diuji, dan responsif. ViewModel mengelola operasi data yang kompleks dan LiveData menyediakan cara yang aman dan efisien untuk mengamati data dari UI tanpa kebocoran memori, memastikan bahwa antarmuka pengguna selalu menampilkan informasi terbaru.