**LAPORAN PRAKTIKUM**

**PEMROGRAMAN MOBILE**

**MODUL 2**

****

**ANDROID LAYOUT WITH COMPOSE**

**Oleh:**

**Avantio Fierza Patria NIM. 2310817310001**

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS TEKNIK**

**UNIVERSITAS LAMBUNG MANGKURAT**

**APRIL 2025**

# LEMBAR PENGESAHAN

**LAPORAN PRAKTIKUM PEMROGRAMAN Mobile**

**MODUL 2**

Laporan Praktikum Pemrograman Mobile Modul 2: Android Layout With Compose ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Avantio Fierza Patria

NIM : 2310817310001

|  |  |
| --- | --- |
| Menyetujui,  Asisten Praktikum  Muhammad Raka Azwar  NIM. 2210817210012 | Mengetahui,  Dosen Penanggung Jawab Praktikum  Andreyan Rizky Baskara, S.Kom., M.Kom.  NIP. 19930703 201903 01 011 |

# DAFTAR ISI

[LEMBAR PENGESAHAN i](#_Toc195980738)

[DAFTAR ISI ii](#_Toc195980739)

[DAFTAR GAMBAR iii](#_Toc195980740)

[DAFTAR TABEL iv](#_Toc195980741)

[SOAL 1 1](#_Toc195980742)

[A. Source Code menggunakan Jetpack Compose 2](#_Toc195980743)

[B. Source Code menggunakan XML 5](#_Toc195980744)

[C. Output Program 8](#_Toc195980745)

[D. Pembahasan 10](#_Toc195980746)

[E. Tautan GIT 12](#_Toc195980747)

# DAFTAR GAMBAR

[Gambar 1. 1 Gambar Tampilan awal aplikasi 1](#_Toc195980878)

[Gambar 1. 2 Tampilan Pilihan Persentase Tip 2](#_Toc195980879)

[Gambar 1. 3 Tampian Aplikasi Setelah Dijalankan 2](#_Toc195980880)

[Gambar 1. 4 Screenshot Tampilan Awal 8](#_Toc195980881)

[Gambar 1. 5 Screenshot Tampilan Pilihan Tip 9](#_Toc195980882)

[Gambar 1. 6 Screenshot Aplikasi Setelah Dijalankan 10](#_Toc195980883)

# DAFTAR TABEL

[Tabel 1. 1 Source Code Jawaban Soal 1 MainActivity.kt Jetpack Compose 5](#_Toc195979422)

[Tabel 1. 2 Source Code Jawaban Soal 1 MainActivity.kt XML 6](#_Toc195979423)

[Tabel 1. 3 Source Code Jawaban Soal 1 activity\_maint .XML 7](#_Toc195979424)

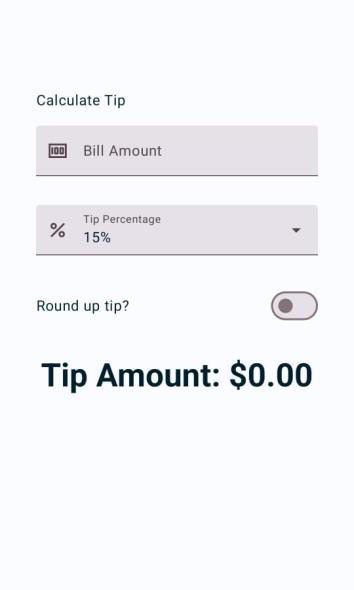
# SOAL 1

Soal Praktikum:

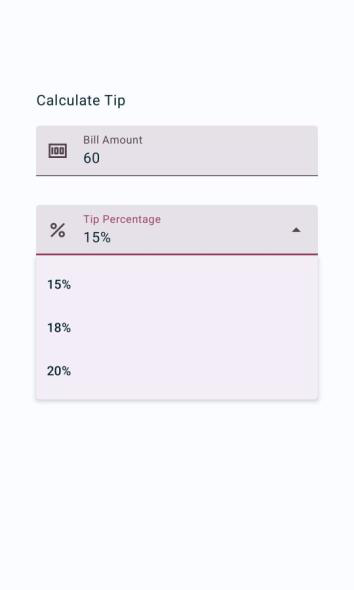
1. Buatlah sebuah aplikasi kalkulator tip menggunakan XML dan Jetpack Compose yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

1. Input biaya layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
2. Pilihan persentase tip: Pengguna dapat memilih persentase tip yang diinginkan.
3. Pengaturan pembulatan tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
4. Tampilan hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.

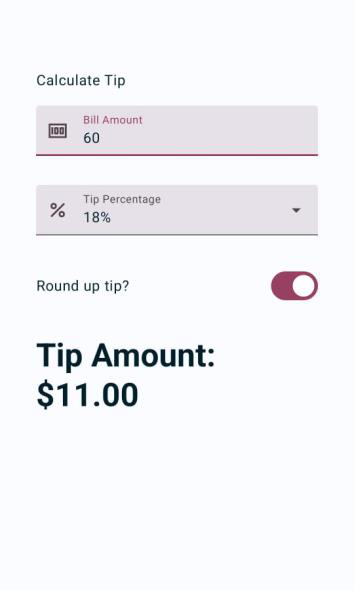
2. Jelaskan perbedaan dari implementasi XML dan Jetpack Compose beserta kelebihan dan kekurangan dari masing-masing implementasi.



Gambar 1. 1 Gambar Tampilan awal aplikasi



Gambar 1. 2 Tampilan Pilihan Persentase Tip



Gambar 1. 3 Tampian Aplikasi Setelah Dijalankan

## Source Code menggunakan Jetpack Compose

**MainActivity.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128 | package com.example.kalkulator  import android.os.Bundle  import androidx.activity.ComponentActivity  import androidx.activity.compose.setContent  import androidx.compose.foundation.layout.\*  import androidx.compose.foundation.text.KeyboardOptions  import androidx.compose.material3.\*  import androidx.compose.runtime.\*  import androidx.compose.ui.Alignment  import androidx.compose.ui.Modifier  import androidx.compose.ui.text.input.KeyboardType  import androidx.compose.ui.unit.dp  import com.example.kalkulator.ui.theme.KalkulatorTheme  import kotlin.math.ceil  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContent {  KalkulatorTheme {  Surface(  modifier = Modifier.fillMaxSize(),  color = MaterialTheme.colorScheme.background  ) {  TipCalculatorCompose()  }  }  }  }  }  @OptIn(ExperimentalMaterial3Api::class)  @Composable  fun TipCalculatorCompose() {  var cost by remember { mutableStateOf("") }  var tipPercent by remember { mutableStateOf(15) }  var roundUp by remember { mutableStateOf(false) }  val tip = calculateTip(cost.toDoubleOrNull() ?: 0.0, tipPercent, roundUp)  Column(  modifier = Modifier  .padding(16.dp)  .fillMaxWidth()  ) {  Text("Kalkulator Tip", style = MaterialTheme.typography.headlineSmall)  Spacer(modifier = Modifier.height(16.dp))  OutlinedTextField(  value = cost,  onValueChange = { cost = it },  label = { Text("Biaya Layanan") },  keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),  modifier = Modifier.fillMaxWidth()  )  Spacer(modifier = Modifier.height(16.dp))  var expanded by remember { mutableStateOf(false) }  val tipOptions = listOf(10, 15, 20, 25)  ExposedDropdownMenuBox(  expanded = expanded,  onExpandedChange = { expanded = !expanded }  ) {  OutlinedTextField(  value = "$tipPercent%",  onValueChange = {},  readOnly = true,  label = { Text("Persentase Tip") },  trailingIcon = { ExposedDropdownMenuDefaults.TrailingIcon(expanded) },  modifier = Modifier.menuAnchor().fillMaxWidth()  )  ExposedDropdownMenu(  expanded = expanded,  onDismissRequest = { expanded = false }  ) {  tipOptions.forEach { percent ->  DropdownMenuItem(  text = { Text("$percent%") },  onClick = {  tipPercent = percent  expanded = false  }  )  }  }  }  Spacer(modifier = Modifier.height(16.dp))  Row(  verticalAlignment = Alignment.CenterVertically,  modifier = Modifier.fillMaxWidth()  ) {  Text("Round up tip?", modifier = Modifier.weight(1f))  Switch(  checked = roundUp,  onCheckedChange = { roundUp = it },  colors = SwitchDefaults.colors(  checkedThumbColor = MaterialTheme.colorScheme.primary  )  )  }  Spacer(modifier = Modifier.height(24.dp))  Text(  text = "Jumlah Tip: Rp%.2f".format(tip),  style = MaterialTheme.typography.titleMedium  )  }  }  fun calculateTip(amount: Double, tipPercent: Int, roundUp: Boolean): Double {  var tip = amount \* tipPercent / 100  if (roundUp) tip = ceil(tip)  return tip  } |

Tabel 1. 1 Source Code Jawaban Soal 1 MainActivity.kt Jetpack Compose

## Source Code menggunakan XML

**MainActivity.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | package com.example.diceroller2  import android.os.Bundle  import android.widget.Button  import android.widget.ImageView  import android.widget.TextView  import androidx.appcompat.app.AppCompatActivity  import kotlin.random.Random  class MainActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  // Ambil referensi dari UI  val dice1Image: ImageView = findViewById(R.id.dice1Image)  val dice2Image: ImageView = findViewById(R.id.dice2Image)  val btnRoll: Button = findViewById(R.id.btnRoll)  val tvMessage: TextView = findViewById(R.id.tvMessage)  btnRoll.setOnClickListener {  val dice1Value = Random.nextInt(1, 7)  val dice2Value = Random.nextInt(1, 7)  // Update gambar dadu  dice1Image.setImageResource(getDiceImage(dice1Value))  dice2Image.setImageResource(getDiceImage(dice2Value))  // Update pesan berdasarkan hasil  tvMessage.text = if (dice1Value == dice2Value) {  "🥳 Selamat, anda dapat dadu double! 🥳"  } else {  "🥺 Anda belum beruntung!"  }  }  }  private fun getDiceImage(value: Int): Int {  return when (value) {  1 -> R.drawable.dice\_1  2 -> R.drawable.dice\_2  3 -> R.drawable.dice\_3  4 -> R.drawable.dice\_4  5 -> R.drawable.dice\_5  6 -> R.drawable.dice\_6  else -> R.drawable.dice\_1  }  }  } |

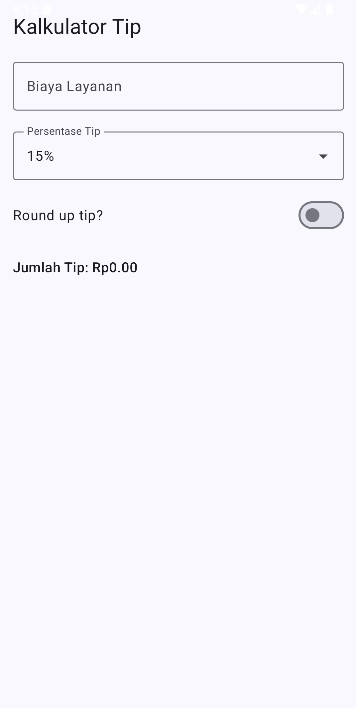
Tabel 1. 2 Source Code Jawaban Soal 1 MainActivity.kt XML

**activity\_main.xml**

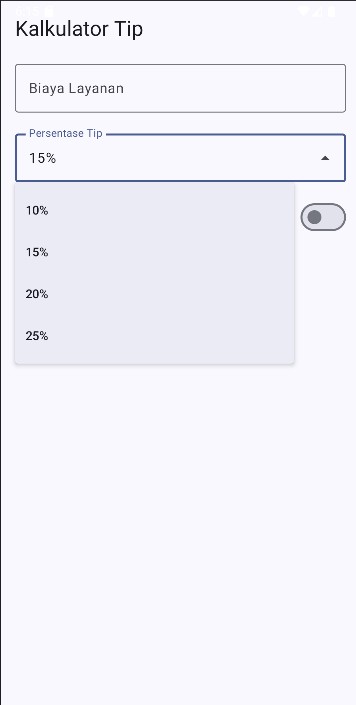
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:padding="16dp"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <TextView  android:text="Kalkulator Tip"  android:textSize="24sp"  android:textStyle="bold"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content" />  <EditText  android:id="@+id/costInput"  android:hint="Biaya Layanan"  android:inputType="numberDecimal"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"/>  <Spinner  android:id="@+id/tipSpinner"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp" />  <LinearLayout  android:orientation="horizontal"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:gravity="center\_vertical">  <TextView  android:text="Round up tip?"  android:layout\_width="0dp"  android:layout\_weight="1"  android:layout\_height="wrap\_content"/>  <Switch  android:id="@+id/roundUpSwitch"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:thumbTint="@android:color/white"/>  </LinearLayout>  <TextView  android:id="@+id/tipResult"  android:text="Jumlah Tip: Rp0.00"  android:textSize="18sp"  android:textStyle="bold"  android:layout\_marginTop="24dp"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"/>  </LinearLayout> |

Tabel 1. 3 Source Code Jawaban Soal 1 activity\_maint .XML

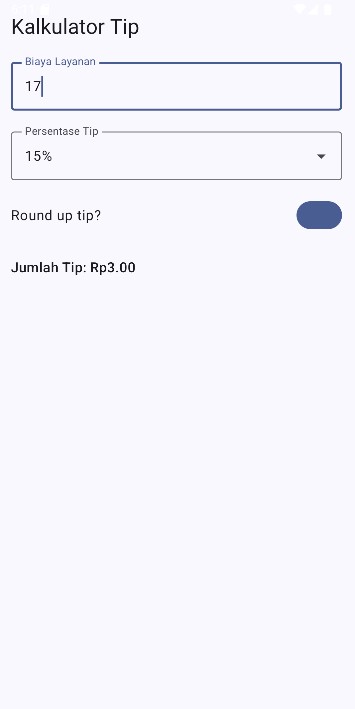
## Output Program



Gambar 1. 4 Screenshot Tampilan Awal



Gambar 1. 5 Screenshot Tampilan Pilihan Tip



Gambar 1. 6 Screenshot Aplikasi Setelah Dijalankan

## Pembahasan

**MainActivity.kt Jetpack Compose:**

1. **MainActivity**

Di *MainActivity*, kita menggunakan komponen *ComponentActivity* karena Compose tidak lagi menggunakan layout XML. Di dalam fungsi *onCreate*, kita langsung menetapkan konten UI menggunakan fungsi *setContent*.

Seluruh tampilan aplikasi dibungkus dalam *Surface*, dan di dalamnya kita memanggil fungsi *TipCalculatorCompose(),* yaitu UI utama kita.

1. **Fungsi** *TipCalculatorCompose()*

Fungsi ini berisi seluruh antarmuka pengguna. Kita mendefinisikan beberapa state:

* *cost* menyimpan input biaya layanan dari pengguna.
* *tipPercent* menyimpan persentase tip yang dipilih.
* *roundUp* menyimpan status switch, apakah pengguna ingin membulatkan tip ke atas.

State ini memungkinkan UI untuk secara otomatis merespons perubahan nilai yang dilakukan pengguna.

1. **Input Biaya Layanan**

Bagian ini menggunakan OutlinedTextField, yaitu versi Compose dari input teks yang familiar di Material Design. Ketika pengguna mengetik, nilai cost akan berubah, dan tip akan dihitung ulang.

1. **Dropdown Pilihan Tip** **(*Spinner*)**

Kita menggunakan ExposedDropdownMenuBox, yang fungsinya mirip seperti Spinner di XML. Pengguna bisa mengetuknya untuk memilih persentase tip (10%, 15%, 20%, atau 25%). Saat dipilih, nilai tipPercent diperbarui.

1. **Switch "Round Up Tip"**

Untuk opsi membulatkan ke atas, kita menggunakan Switch() yang diletakkan dalam Row, berdampingan dengan teks deskriptif. Jika pengguna mengaktifkan switch ini, nilai tip akan dibulatkan ke atas menggunakan ceil().

1. **Tampilan Hasil Tip**

Di bagian bawah, kita menampilkan hasil perhitungan tip dalam format "Rp...", dan tampilannya akan berubah secara otomatis ketika input, persentase, atau switch diubah.

1. **Fungsi calculateTip()**

Fungsi ini menerima tiga parameter: jumlah biaya, persentase tip, dan status pembulatan. Fungsi ini melakukan perhitungan tip, dan jika roundUp bernilai true, maka nilai tip akan dibulatkan ke atas.

**MainActivity.kt XML:**

1. **activity\_main.xml**

File ini berisi antarmuka pengguna menggunakan pendekatan tradisional. Komponen-komponen utama antara lain:

* EditText: Tempat pengguna memasukkan biaya layanan.
* Spinner: Digunakan untuk memilih persentase tip.
* Switch: Digunakan untuk memilih apakah ingin membulatkan ke atas.
* TextView: Menampilkan hasil tip.

View: Sebagai pemisah visual (divider) di bawah spinner agar tampilan lebih rapi.

1. **MainActivity.kt**

Di sini kita menggunakan pendekatan View-based (lama). Di dalam onCreate, kita menghubungkan semua komponen dari XML ke Kotlin menggunakan findViewById.

1. **Inisialisasi Spinner**

Kita menyiapkan daftar persentase tip dalam bentuk list (tipOptions), lalu memasangnya ke spinner menggunakan ArrayAdapter. Dengan begitu, spinner akan menampilkan pilihan seperti 10%, 15%, dst.

1. **Fungsi updateTip()**

Fungsi ini adalah inti dari logika kalkulasi. Ia membaca nilai biaya, persentase tip yang dipilih dari spinner, dan status dari switch. Kemudian ia menghitung tip dan memperbarui teks di TextView.

1. **Event Listener**

Agar aplikasi merespons secara real-time, kita menambahkan listener:

* Pada EditText, kita menggunakan TextWatcher agar setiap kali pengguna mengetik, tip langsung dihitung ulang.
* Pada Switch, kita mendengarkan perubahan status.
* Pada Spinner, kita merespons ketika pengguna memilih persentase baru.

Semua listener tersebut akan memanggil fungsi updateTip() agar tampilan hasil selalu sesuai input.

Perbedaan dari implementasi XML dan Jetpack Compose beserta kelebihan dan kekurangan dari masing-masing implementasi

XML + View System

XML menggunakan pendekatan deklaratif statis di mana elemen UI ditulis dalam file XML terpisah dan dikaitkan dengan logika di file Kotlin/Java melalui kode imperative. Ini adalah pendekatan konvensional dan telah digunakan sejak awal Android.

Kelebihan:

* Sudah matang dan stabil, digunakan sejak Android awal.
* Lebih familiar bagi developer lama.
* Banyak dokumentasi, tutorial, dan komunitas luas.
* Terpisah antara desain dan logika (bisa didesain pakai layout editor).

Kekurangan:

* Perlu findViewById() atau ViewBinding → verbose.
* Sulit untuk membuat UI yang dinamis atau reaktif.
* Penambahan fitur atau animasi lebih kompleks.
* Struktur hierarki UI bisa rumit dan lambat saat render.

Jetpack Compose

Jetpack Compose adalah framework UI declarative modern berbasis Kotlin, yang memungkinkan kita membangun antarmuka UI secara langsung di dalam file Kotlin, menyatukan logika dan tampilan dalam satu tempat. Komponen UI di Compose bisa langsung merefleksikan perubahan data secara otomatis (reactive).

Kelebihan:

* Lebih modern, efisien, dan ringkas.
* UI dan logika digabung → mudah di-maintain.
* Reaktif → otomatis merender ulang saat state berubah.
* Mendukung tema dinamis, animasi, dan custom UI dengan lebih mudah.
* Cocok untuk arsitektur modern (MVVM, Clean Architecture).

Kekurangan:

* Butuh penyesuaian untuk developer lama.
* Beberapa komponen UI belum sekomplit XML (tapi terus berkembang).
* Dokumentasi dan tooling masih dalam tahap berkembang.
* Kurang cocok untuk proyek Android lama (pre-Jetpack).

## Tautan GIT

**https://github.com/gr1ff0m/Pemrograman-Mobile-Praktikum**