

**LAPORAN PRAKTIKUM  
PEMROGRAMAN MOBILE  
MODUL 2**



**ANDROID LAYOUT WITH COMPOSE**

**Oleh:**

**Muhammad Firas**

**NIM. 2210817110014**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS LAMBUNG MANGKURAT  
MEI 2025**

**LEMBAR PENGESAHAN**  
**LAPORAN PRAKTIKUM PEMROGRAMAN I**  
**MODUL 2**

Laporan Praktikum Pemrograman Mobile Modul 1: Android Layout with Compose ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Muhammad Firas  
NIM : 2210817110014

Menyetujui,  
Asisten Praktikum

Mengetahui,  
Dosen Penanggung Jawab Praktikum

Muhammad Raka Azwar  
NIM. 2210817210012

Andreyan Rizky Baskara, S.Kom., M.Kom.  
NIP. 19930703 201903 01 011

## DAFTAR ISI

LEMBAR PENGESAHAN .....	2
DAFTAR ISI.....	3
DAFTAR GAMBAR .....	4
DAFTAR TABEL.....	5
SOAL 1 .....	6
A.    Source Code .....	8
B.    Output Program.....	15
C.    Pembahasan.....	16
D.    Tautan Git.....	20

## DAFTAR GAMBAR

Gambar 1. Tampilan Awal Aplikasi.....	7
Gambar 2. Tampilan Pilihan Persentase Tip.....	7
Gambar 3. Tampilan Aplikasi Setelah Dijalankan.....	8
Gambar 4. Screenshot Hasil Jawaban Soal 1 XML .....	15
Gambar 5. Screenshot Hasil Jawaban Soal Jetpack Compose.....	15

## DAFTAR TABEL

Tabel 1. Source Code Jawaban Soal 1 MainActivity XML.....	8
Tabel 2. Source Code Jawaban Soal 1 activity_main XML .....	9
Tabel 3. Source Code Jawaban Soal 1 MainActivity Jetpack Compose .....	12

## 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:
  - a. Input biaya layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
  - b. Pilihan persentase tip: Pengguna dapat memilih persentase tip yang diinginkan.
  - c. Pengaturan pembulatan tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
  - d. 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.

Calculate Tip

Bill Amount

% Tip Percentage  
15%

Round up tip? ☐

**Tip Amount: \$0.00**

*Gambar 1. Tampilan Awal Aplikasi*

Calculate Tip

Bill Amount  
60

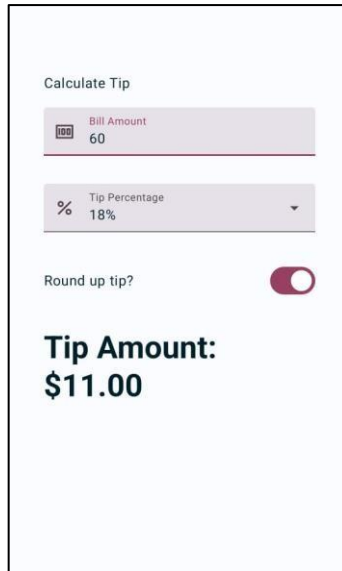
% Tip Percentage  
15%

15%

18%

20%

*Gambar 2. Tampilan Pilihan Persentase Tip*



*Gambar 3. Tampilan Aplikasi Setelah Dijalankan*

## A. Source Code MainActivity.kt

*Tabel 1. Source Code Jawaban Soal 1 MainActivity XML*

```

1 package com.example.xmltipcalculator
2
3 import android.os.Bundle
4 import android.text.Editable
5 import android.text.TextWatcher
6 import android.widget.*
7 import androidx.appcompat.app.AppCompatActivity
8 import
9     com.google.android.material.textfield.MaterialAutoCompleteTextView
10    import kotlin.math.ceil
11
12 class MainActivity : AppCompatActivity() {
13     private lateinit var billAmountInput: EditText
14     private lateinit var tipDropdown: MaterialAutoCompleteTextView
15     private lateinit var roundUpSwitch: Switch
16     private lateinit var tipResultText: TextView
17
18     override fun onCreate(savedInstanceState: Bundle?) {
19         super.onCreate(savedInstanceState)
20         setContentView(R.layout.activity_main)
21
22         // Initialize views
23         billAmountInput = findViewById(R.id.billAmountInput)
24         tipDropdown = findViewById(R.id.tipPercentageDropdown)
25         roundUpSwitch = findViewById(R.id.roundUpSwitch)
26         tipResultText = findViewById(R.id.tipResultText)

```



27	
28	// Set up dropdown
29	val tipOptions = listOf("10%", "15%", "18%", "20%")
30	val adapter = ArrayAdapter(this,
	android.R.layout.simple_list_item_1, tipOptions)
31	tipDropdown.setAdapter(adapter)
32	tipDropdown.setText(tipOptions[1], false) // Default to
	15%
33	
34	// Listeners
35	billAmountInput.addTextChangedListener(object :
	TextWatcher {
36	override fun afterTextChanged(s: Editable?) =
	updateTip()
37	override fun beforeTextChanged(s: CharSequence?,
	start: Int, count: Int, after: Int) {}
38	override fun onTextChanged(s: CharSequence?, start:
	Int, before: Int, count: Int) {}
39	})
40	
41	tipDropdown.setOnItemClickListener { _, _, _, _ ->
	updateTip() }
42	
43	roundUpSwitch.setOnCheckedChangeListener { _, _ ->
	updateTip() }
44	
45	updateTip()
46	}
47	
48	private fun updateTip() {
49	val cost =
	billAmountInput.text.toString().toDoubleOrNull() ?: 0.0
50	val percent = tipDropdown.text.toString().replace("%",
51	"").toIntOrNull() ?: 0
52	val roundUp = roundUpSwitch.isChecked
53	
54	var tip = cost * percent / 100
55	if (roundUp) tip = ceil(tip)
56	
57	tipResultText.text = "Tip Amount: \$%.2f".format(tip)
58	}
59	}

## activity\_main.xml

Tabel 2. Source Code Jawaban Soal 1 activity\_main XML

1	<?xml version="1.0" encoding="utf-8"?>
2	<ScrollView
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"

```

4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:padding="24dp"
7      android:background="#F5F5F5">
8
9      <LinearLayout
10         android:layout_width="match_parent"
11         android:layout_height="wrap_content"
12         android:orientation="vertical">
13
14         <!-- Title -->
15         <TextView
16             android:id="@+id/titleText"
17             android:layout_width="wrap_content"
18             android:layout_height="wrap_content"
19             android:text="Calculate Tip"
20             android:textSize="22sp"
21             android:textStyle="bold"
22             android:textColor="#1C1B1F"
23             android:layout_marginBottom="24dp" />
24
25         <!-- Bill Amount Input -->
26         <com.google.android.material.textfield.TextInputLayout
27             android:layout_width="match_parent"
28             android:layout_height="wrap_content"
29             android:hint="Bill Amount"
30
31             style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedB
32             ox"
33
34             app:startIconDrawable="@drawable/ic_money"
35             app:startIconTint="@android:color/transparent"
36             app:hintTextColor="#888888"
37             android:layout_marginBottom="16dp">
38
39         <com.google.android.material.textfield.TextInputEditText
40             android:id="@+id/billAmountInput"
41             android:layout_width="match_parent"
42             android:layout_height="wrap_content"
43             android:inputType="numberDecimal"
44             android:textColor="#000000" />
45         </com.google.android.material.textfield.TextInputLayout>
46
47         <!-- Tip Percentage Dropdown -->
48         <com.google.android.material.textfield.TextInputLayout
49             android:layout_width="match_parent"
50             android:layout_height="wrap_content"
51             android:hint="Tip Percentage"
52
53             style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedB
54             ox"
55
56             app:startIconDrawable="@drawable/ic_percent"

```

```

51         app:endIconMode="dropdown_menu"
52         app:hintTextColor="#888888"
53         android:layout_marginBottom="16dp">
54
55     <com.google.android.material.textfield.MaterialAutoCompleteTextVi
ew
56         android:id="@+id/tipPercentageDropdown"
57         android:layout_width="match_parent"
58         android:layout_height="wrap_content"
59         android:focusable="false"
60         android:cursorVisible="false"
61         android:inputType="none"
62         android:textColorHint="#888888"
63         android:textColor="#000000" />
64     </com.google.android.material.textfield.TextInputLayout>
65
66     <!-- Round up tip switch -->
67     <LinearLayout
68         android:layout_width="match_parent"
69         android:layout_height="wrap_content"
70         android:orientation="horizontal"
71         android:gravity="center_vertical"
72         android:layout_marginBottom="16dp">
73
74         <TextView
75             android:layout_width="wrap_content"
76             android:layout_height="wrap_content"
77             android:text="Round up tip?"
78             android:textColor="#1C1B1F" />
79
80         <Switch
81             android:id="@+id/roundUpSwitch"
82             android:layout_width="269dp"
83             android:layout_height="wrap_content"
84             android:layout_marginStart="8dp"
85             android:thumbTint="#9C27B0"
86             android:trackTint="#E1BEE7" />
87     </LinearLayout>
88
89     <!-- Tip Result -->
90     <TextView
91         android:id="@+id/tipResultText"
92         android:layout_width="match_parent"
93         android:layout_height="wrap_content"
94         android:gravity="start"
95         android:text="Tip Amount: $0.00"
96         android:textColor="#000000"
97         android:textSize="30sp"
98         android:textStyle="bold" />
99     </LinearLayout>
100 </ScrollView>

```

## MainActivity.kt

Tabel 3. Source Code Jawaban Soal 1 MainActivity Jetpack Compose

```
1 package com.example.jetpackcomposetipcalculator
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import androidx.compose.foundation.interaction.DragInteraction
7 import androidx.compose.foundation.layout.*
8 import androidx.compose.foundation.text.KeyboardOptions
9 import androidx.compose.material3.*
10 import androidx.compose.runtime.*
11 import androidx.compose.ui.Alignment
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.graphics.Color
14 import androidx.compose.ui.text.font.FontWeight
15 import androidx.compose.ui.text.input.KeyboardType
16 import androidx.compose.ui.text.style.LineHeightStyle
17 import androidx.compose.ui.tooling.preview.Preview
18 import androidx.compose.ui.unit.dp
19 import androidx.compose.ui.unit.sp
20 import kotlin.math.ceil
21
22 class MainActivity : ComponentActivity() {
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         setContent {
26             MaterialTheme {
27                 TipCalculatorApp()
28             }
29         }
30     }
31 }
32
33 @Composable
34 fun TipCalculatorApp() {
35     var billAmount by remember { mutableStateOf("") }
36     var tipPercent by remember { mutableStateOf("15") }
37     var roundUp by remember { mutableStateOf(false) }
38
39     val tip = calculateTip(billAmount,
40 tipPercent.toIntOrNull() ?: 0, roundUp)
41
42     Column(
43         modifier = Modifier
44             .fillMaxSize()
45             .padding(24.dp),
46         verticalArrangement = Arrangement.spacedBy(16.dp),
47         horizontalAlignment = Alignment.CenterHorizontally
```

```

47     ) {
48         Text(
49             "Calculate Tip",
50             fontSize = 22.sp,
51             modifier = Modifier.align(Alignment.Start))
52
53         OutlinedTextField(
54             value = billAmount,
55             onChange = { billAmount = it },
56             label = { Text("Bill Amount") },
57             keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Number),
58             modifier = Modifier.fillMaxWidth()
59         )
60
61         TipDropdown(
62             selected = tipPercent,
63             onChange = { tipPercent = it }
64         )
65
66         Row(
67             modifier = Modifier.fillMaxWidth(),
68             verticalAlignment = Alignment.CenterVertically,
69             horizontalArrangement = Arrangement.SpaceBetween
70         ) {
71             Text("Round up tip?")
72             Switch(
73                 checked = roundUp,
74                 onCheckedChange = { roundUp = it }
75             )
76         }
77
78         Text(
79             text = "Tip Amount: $${"%.2f".format(tip)}",
80             fontSize = 30.sp,
81             color = Color.Black,
82             fontWeight = FontWeight.Bold,
83             modifier = Modifier.align(Alignment.Start)
84         )
85     }
86 }
87
88 @OptIn(ExperimentalMaterial3Api::class)
89 @Composable
90 fun TipDropdown(
91     selected: String,
92     onChange: (String) -> Unit
93 ) {
94     val options = listOf("10", "15", "18", "20")
95     var expanded by remember { mutableStateOf(false) }
96
97     ExposedDropdownMenuBox(

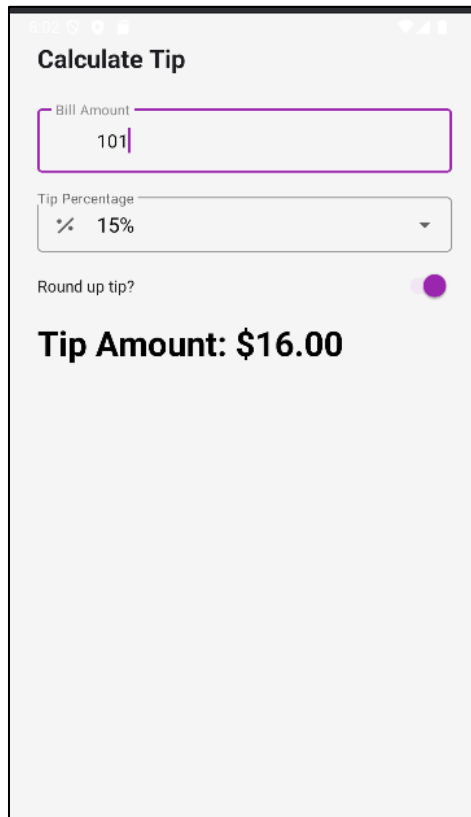
```

```

98         expanded = expanded,
99         onExpandedChange = { expanded = !expanded }
100     ) {
101         TextField(
102             readOnly = true,
103             value = "$selected%",
104             onValueChange = {},
105             label = { Text("Tip Percentage") },
106             trailingIcon = {
107
108                 ExposedDropDownMenuDefaults.TrailingIcon(expanded = expanded)
109             },
110             colors =
111                 ExposedDropDownMenuDefaults.textFieldColors(),
112             modifier = Modifier
113                 .menuAnchor()
114                 .fillMaxWidth()
115         )
116
117         ExposedDropDownMenu(
118             expanded = expanded,
119             onDismissRequest = { expanded = false }
120         ) {
121             options.forEach { percent ->
122                 DropdownMenuItem(
123                     text = { Text("$percent%") },
124                     onClick = {
125                         onChange(percent)
126                         expanded = false
127                     }
128                 )
129             }
130         }
131     }
132
133 fun calculateTip(amount: String, percent: Int, roundUp:
134 Boolean): Double {
135     val cost = amount.toDoubleOrNull() ?: return 0.0
136     var tip = cost * percent / 100
137     if (roundUp) tip = ceil(tip)
138     return tip
139 }
140
141 @Preview(showBackground = true, name = "Tip Calculator
142 Preview")
143 @Composable
144 fun TipCalculatorPreview() {
145     MaterialTheme {
146         TipCalculatorApp()
147     }
148 }

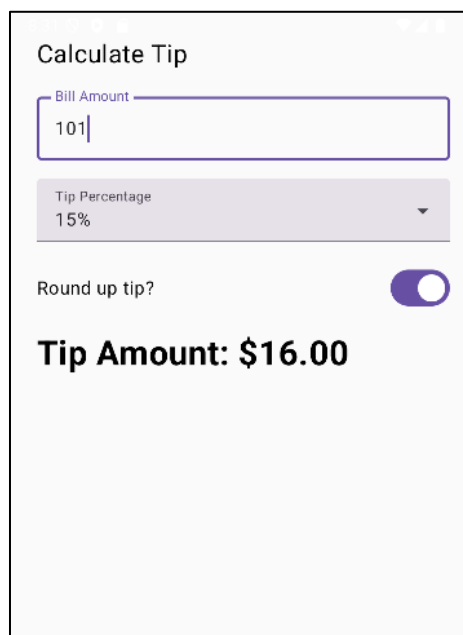
```

## B. Output Program



The screenshot shows a mobile application titled "Calculate Tip". It features a text input field for "Bill Amount" containing the value "101". Below it is a dropdown menu for "Tip Percentage" set to "15%". A toggle switch for "Round up tip?" is turned on. The result, "Tip Amount: \$16.00", is displayed in a large, bold font.

Gambar 4. Screenshot Hasil Jawaban Soal 1 XML



The screenshot shows a mobile application titled "Calculate Tip". It features a text input field for "Bill Amount" containing the value "101". Below it is a dropdown menu for "Tip Percentage" set to "15%". A toggle switch for "Round up tip?" is turned on. The result, "Tip Amount: \$16.00", is displayed in a large, bold font.

Gambar 5. Screenshot Hasil Jawaban Soal Jetpack Compose

## C. Pembahasan

### 1. Pembahasan Praktikum:

#### MainActivity.kt (XML):

Pada baris [1], `package com.example.xmltipcalculator` berfungsi untuk mendefinisikan namespace atau ruang nama dari aplikasi ini, yang digunakan untuk mengatur struktur dan lokasi file di dalam proyek Android. Pada baris [3–7], berbagai library Android seperti `android.os.Bundle`, `android.widget.*`, dan `androidx.appcompat.app.AppCompatActivity` di-*import* agar fitur-fitur seperti `TextView`, `EditText`, `Switch`, dan lainnya bisa digunakan dalam kode. Library `kotlin.math.ceil` juga di-*import* pada baris [8] untuk membulatkan angka ke atas saat menghitung tip.

Pada baris [11], kelas `MainActivity` dideklarasikan sebagai turunan dari `AppCompatActivity`, yang merupakan kelas dasar untuk semua *activity* di Android menggunakan komponen Material Design. Pada baris [12–16], beberapa variabel dideklarasikan menggunakan keyword `lateinit`, yang berarti variabel tersebut akan di-*initialize* nanti. Variabel tersebut mencakup `billAmountInput` untuk input tagihan, `tipDropDown` untuk memilih persentase tip, `roundUpSwitch` untuk opsi membulatkan, dan `tipResultText` untuk menampilkan hasil perhitungan tip.

Pada baris [18], fungsi `onCreate` dimulai, yaitu fungsi yang pertama kali dipanggil saat *activity* dimuat. Pada baris [19], `super.onCreate(savedInstanceState)` memanggil fungsi `onCreate` dari superclass untuk memastikan semua proses dasar dijalankan. Pada baris [20], `setContentView(R.layout.activity_main)` digunakan untuk mengatur tampilan *activity* berdasarkan file layout XML yang bernama `activity_main.xml`.

Pada baris [22–26], masing-masing elemen antarmuka di-*initialize* menggunakan `findViewById()` dengan ID yang sesuai dari layout XML. Misalnya, `billAmountInput` dihubungkan ke `R.id.billAmountInput`, dan seterusnya. Pada baris [29], daftar pilihan tip berupa `tipOptions` dideklarasikan sebagai list string berisi "10%", "15%", "18%", dan "20%". Pada baris [30], `ArrayAdapter` dibuat untuk menghubungkan list tersebut ke `tipDropDown`, dan pada baris [32], pilihan default yang ditampilkan adalah "15%".

Pada baris [35–38], `billAmountInput` diberikan *listener* bernama `TextWatcher` agar setiap perubahan input langsung memicu fungsi `updateTip()`. Listener ini memiliki tiga metode, namun hanya `afterTextChanged` yang diisi dengan logika untuk menghitung tip. Pada baris [41], dropdown persentase tip diberikan *listener* menggunakan `setOnItemClickListener` agar saat user memilih nilai tip, fungsi `updateTip()` juga dijalankan. Pada baris [43], `roundUpSwitch` juga diberi listener dengan `setOnCheckedChangeListener` agar saat switch dinyalakan atau dimatikan, nilai tip akan diperbarui.



Pada baris [45], fungsi `updateTip()` dipanggil sekali agar hasil default muncul sejak awal. Pada baris [48], fungsi `updateTip()` didefinisikan untuk menghitung nilai tip berdasarkan input pengguna. Pada baris [49], input nilai tagihan dikonversi menjadi `Double`, dan jika gagal akan dianggap sebagai `0.0`. Pada baris [50], nilai persentase tip diambil dari dropdown, dibuang simbol `%`, lalu dikonversi ke `Int`. Jika gagal akan menjadi `0`. Pada baris [51], nilai boolean `roundUp` akan bernilai `true` jika switch aktif.

Pada baris [53], perhitungan tip dilakukan dengan mengalikan `cost` dan `percent`, lalu dibagi `100`. Jika `roundUp` bernilai `true`, maka nilai `tip` akan dibulatkan ke atas menggunakan fungsi `ceil`. Pada baris [56], hasil tip ditampilkan ke pengguna dalam `tipResultText`, dengan format `$.2f` agar hasil selalu dua angka di belakang koma.

### **activity\_main.xml (XML):**

Pada baris [1], `<?xml version="1.0" encoding="utf-8"?>` merupakan deklarasi XML yang menyatakan versi XML yang digunakan yaitu versi 1.0, serta encoding-nya adalah UTF-8, yang memungkinkan penggunaan karakter internasional. Pada baris [2-7], tag `<ScrollView>` digunakan untuk membuat tampilan yang dapat discroll secara vertikal, sehingga jika isi konten melebihi ukuran layar, pengguna tetap dapat menggulir ke bawah. `ScrollView` ini memiliki atribut `layout_width` dan `layout_height` yang diatur ke `match_parent` agar memenuhi seluruh layar, dengan padding `24dp` dan warna latar belakang abu-abu muda `#F5F5F5`.

Pada baris [9-12], `<LinearLayout>` digunakan untuk menampung seluruh elemen UI di dalam `ScrollView`. Atribut `orientation="vertical"` menunjukkan bahwa elemen-elemen di dalam `LinearLayout` akan ditata dari atas ke bawah. `layout_width` diatur ke `match_parent` agar selebar layar, sementara `layout_height` adalah `wrap_content` agar hanya setinggi kontennya.

Pada baris [15-23], `<TextView>` digunakan untuk menampilkan judul aplikasi yaitu "Calculate Tip". Komponen ini memiliki ukuran teks `22sp`, tebal (`bold`), warna teks `#1C1B1F`, dan margin bawah `24dp` untuk memberi jarak dari elemen berikutnya.

Pada baris [26-34], `<TextInputLayout>` digunakan sebagai pembungkus input teks untuk jumlah tagihan (Bill Amount). Komponen ini menggunakan style `OutlinedBox` dari Material Components untuk memberikan efek kotak pada input. Ia juga dilengkapi dengan ikon `ic_money` di awal, namun diberi `tint` transparan agar tidak tampak. Atribut `hint` diatur ke "Bill Amount", dan margin bawahnya adalah `16dp`. Di dalamnya, pada baris [36-41], terdapat `<TextInputEditText>` yang memiliki `id="billAmountInput"`, digunakan untuk menerima input angka desimal dari pengguna, dengan warna teks hitam.

Pada baris [45-53], `<TextInputLayout>` berikutnya digunakan untuk membuat dropdown pemilihan persentase tip (Tip Percentage). Sama seperti sebelumnya, komponen ini juga menggunakan style `OutlinedBox`, tapi kali ini dengan ikon persen `ic_percent` dan `endIconMode="dropdown_menu"` agar dropdown bisa dibuka. Hint-nya adalah "Tip Percentage", dan margin bawah `16dp`. Pada baris [55-63], di dalamnya terdapat

`<MaterialAutoCompleteTextView>` dengan `id="tipPercentageDropdown"` yang akan menampilkan daftar persentase tip. Atribut `focusable` dan `cursorVisible` diatur ke `false` agar tidak bisa diketik manual. Warna hint-nya adalah abu-abu dan warna teksnya hitam.

Pada baris [67-72], `<LinearLayout>` horizontal digunakan untuk menampung teks dan switch untuk opsi membulatkan nilai tip. `gravity="center_vertical"` memastikan bahwa elemen-elemen di dalamnya rata secara vertikal. Margin bawahnya diatur ke `16dp`. Pada baris [74-78], terdapat `<TextView>` dengan teks "Round up tip?" dan warna teks hitam. Pada baris [80-86], `<Switch>` dengan `id="roundUpSwitch"` digunakan untuk memberikan opsi pembulatan. Switch ini memiliki lebar `269dp`, margin kiri `8dp`, warna thumb ungu `#9C27B0`, dan track warna ungu muda `#E1BEE7`.

Pada baris [90-98], `<TextView>` terakhir digunakan untuk menampilkan hasil kalkulasi tip. `id`-nya adalah `tipResultText`, `layout_width`-nya `match_parent`, dan `layout_height`-nya `wrap_content`. Teks default-nya adalah "Tip Amount: \$0.00", dengan ukuran teks `30sp`, teks tebal (bold), warna hitam, dan teks diratakan ke kiri menggunakan `gravity="start"`.

Pada baris [99], `</LinearLayout>` menutup tag layout utama yang menampung semua elemen. Pada baris [100], `</ScrollView>` menutup tag `ScrollView` dan mengakhiri struktur layout dari `activity_main.xml`.

### **MainActivity.kt (Jetpack Compose) :**

Pada baris [1], `package com.example.jetpackcomposetipcalculator` mendefinisikan nama paket dari aplikasi ini, yang berguna untuk mengorganisir kode dan mencegah konflik nama. Pada baris [3–20], `import` digunakan untuk mengimpor berbagai fungsi dan komponen dari pustaka Android dan Jetpack Compose seperti `ComponentActivity`, `setContent`, `Text`, `OutlinedTextField`, `Modifier`, `Color`, `FontWeight`, dan lain-lain agar bisa digunakan dalam file ini. Pada baris [22], `class MainActivity : ComponentActivity()` mendeklarasikan kelas utama aplikasi yang mewarisi dari `ComponentActivity`, yang merupakan dasar dari aktivitas Jetpack Compose. Pada baris [23], fungsi `onCreate()` merupakan titik awal ketika aktivitas dijalankan. Di dalamnya, pada baris [24], `super.onCreate(savedInstanceState)` memanggil implementasi `onCreate` dari superclass-nya. Pada baris [25], `setContent` digunakan untuk menyetel tampilan UI dengan Compose. Di dalamnya, pada baris [26], `MaterialTheme` digunakan untuk menerapkan tema Material Design pada seluruh UI. Pada baris [27], `TipCalculatorApp()` dipanggil untuk merender seluruh isi aplikasi kalkulator tip.

Pada baris [34], fungsi `TipCalculatorApp()` adalah fungsi Compose utama yang menampilkan UI kalkulator. Pada baris [35–37], tiga buah `mutableStateOf` dideklarasikan menggunakan `remember` untuk menyimpan input pengguna: `billAmount` untuk jumlah tagihan, `tipPercent` untuk persentase tip, dan `roundUp` sebagai boolean apakah hasil tip dibulatkan. Pada baris [39], `val tip` menghitung jumlah tip menggunakan fungsi `calculateTip()` dengan memasukkan input dari pengguna.

Pada baris [41-46], digunakan `Column` sebagai layout vertikal yang menyusun elemen-elemen UI dari atas ke bawah. Atribut `modifier` menyetel layout agar memenuhi layar (`fillMaxSize`) dan diberi `padding 24dp`. `verticalArrangement` dan `horizontalAlignment` digunakan untuk mengatur jarak antar elemen dan perataan secara horizontal di tengah. Pada baris [38-51], `Text()` digunakan untuk menampilkan judul "Calculate Tip" dengan ukuran teks `22sp`, dan diposisikan ke kiri menggunakan `align(Alignment.Start)`.

Pada baris [53-58], `OutlinedTextField` digunakan sebagai input untuk jumlah tagihan, dengan label "Bill Amount". `keyboardOptions` diatur agar hanya menerima input berupa angka (`KeyboardType.Number`), dan `modifier.fillMaxWidth()` membuat lebar input selebar layar.

Pada baris [61-63], `TipDropdown()` adalah fungsi `Compose` terpisah yang menampilkan dropdown untuk memilih persentase tip. Parameter `selected` dan `onChange` digunakan untuk mengatur nilai pilihan dan logika saat berubah.

Pada baris [66-69], `Row` digunakan untuk menampilkan teks dan switch dalam satu baris horizontal. `fillMaxWidth()` membuat komponen ini selebar layar. `verticalAlignment` menyelaraskan konten secara vertikal di tengah, dan `horizontalArrangement.SpaceBetween` mendorong konten ke kiri dan kanan layar. Di dalamnya, pada baris [71], `Text("Round up tip?")` menampilkan label teks, dan pada baris [72-74], `Switch` digunakan untuk memberi opsi apakah nilai tip perlu dibulatkan, dengan `checked` mengacu pada state `roundUp`.

Pada baris [78-83], `Text` digunakan untuk menampilkan hasil tip dengan teks "Tip Amount: \$...", ukuran `30sp`, warna hitam, dan font tebal, serta diratakan ke kiri.

Pada baris [90-92], fungsi `TipDropdown()` didefinisikan. Di dalamnya, pada baris [94], dideklarasikan daftar opsi persentase tip (10, 15, 18, 20). Pada baris [95], `expanded` adalah `mutableStateOf` yang menyimpan status apakah dropdown terbuka atau tertutup. Pada baris [97-99], `ExposedDropdownMenuBox` adalah komponen dari `Material3` yang membungkus `TextField` dan `DropdownMenu`. Atribut `onExpandedChange` digunakan untuk mengubah status `expanded`. Pada baris [101-112], `TextField` ditampilkan sebagai kotak dropdown yang hanya bisa dibaca (`readOnly = true`), dengan teks pilihan, label "Tip Percentage", dan ikon panah dropdown di sebelah kanan (`trailingIcon`). `menuAnchor()` dan `fillMaxWidth()` memastikan dropdown menempel pada kotak input dan memenuhi lebar tampilan. Pada baris [115-125], `ExposedDropdownMenu` ditampilkan saat `expanded` bernilai `true`. Di dalamnya, `options.forEach` digunakan untuk menampilkan setiap persentase sebagai `DropdownMenuItem`. Saat salah satu dipilih, `onChange(percent)` dipanggil dan dropdown ditutup (`expanded = false`).

Pada baris [132], fungsi `calculateTip()` mendefinisikan logika untuk menghitung jumlah tip. Parameter `amount` adalah input string jumlah tagihan, `percent` adalah persentase tip, dan `roundUp` menentukan apakah hasil dibulatkan. Pada baris [133], `toDoubleOrNull()`

mengubah string menjadi angka, atau mengembalikan 0.0 jika tidak valid. Pada baris [134-136], tip dihitung dengan mengalikan `cost` dan `percent` dibagi 100. Jika `roundUp == true`, maka `ceil()` digunakan untuk membulatkan hasil ke atas.

Pada baris [141], `@Preview` digunakan untuk menampilkan pratinjau UI di Android Studio. `TipCalculatorPreview()` memanggil `TipCalculatorApp()` di dalam `MaterialTheme` agar bisa dilihat tampilannya sebelum dijalankan di emulator atau perangkat.

## **2. Pembahasan Perbedaan Implementasi XML dan Jetpack Compose:**

Jadi, perbedaan utama antara implementasi XML dan Jetpack Compose dalam pengembangan antarmuka pengguna (UI) Android terletak pada pendekatannya. XML merupakan pendekatan imperatif berbasis file terpisah, di mana layout UI ditulis dalam file XML yang terpisah dari logika program (kode Java/Kotlin). Sementara itu, Jetpack Compose adalah framework deklaratif modern dari Android yang memungkinkan UI dibangun langsung di dalam kode Kotlin tanpa perlu file XML. Ini membuat Jetpack Compose menawarkan integrasi lebih erat antara logika dan tampilan, serta lebih fleksibel dan efisien dalam membuat UI yang dinamis atau kompleks.

Nah, kelebihan XML disini ialah kematangan dan kestabilannya, karena telah digunakan sejak lama dan didukung oleh banyak dokumentasi serta komunitas. Selain itu, XML memisahkan antara tampilan dan logika, yang bisa memudahkan pengelolaan dalam tim besar. Namun, XML bisa terasa bertele-tele dan kurang fleksibel, terutama saat membuat UI yang dinamis, serta membutuhkan banyak boilerplate code.

Sementara, Jetpack Compose menawarkan penulisan UI yang lebih ringkas, reaktif, dan intuitif, terutama dalam membuat UI kompleks yang bergantung pada state. Jetpack Compose juga lebih mudah untuk diuji dan diintegrasikan dengan arsitektur modern seperti MVVM. Namun, kekurangannya adalah karena masih tergolong relatif baru, Jetpack Compose bisa memiliki bug, keterbatasan library pihak ketiga, serta membutuhkan pembiasaan baru bagi developer yang terbiasa dengan XML.

Jadi intinya, XML cocok untuk aplikasi yang stabil dan bersifat legacy, sedangkan Jetpack Compose lebih ideal untuk proyek baru yang membutuhkan fleksibilitas dan efisiensi dalam pengembangan UI modern.

### **D. Tautan Git**

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/MuhammadFiras/Praktikum-Mobile/tree/main>