

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2**



Android Layout With Compose

Oleh:

Muhammad Daffa Musyafa

NIM. 2310817110007

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 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 Daffa Musyafa
NIM : 2310817210007

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 XML.....	7
B. Output Program	14
C. Pembahasan	16
D. Source Code Compose	19
E. Output Program	24
F. Pembahasan	25
Soal 2	27
Tautan Git	27

DAFTAR GAMBAR

Gambar 1 Tampilan Awal Aplikasi.....	6
Gambar 2 Tampilan Pilihan Persentase Tip	7
<i>Gambar 3 Tampilan Aplikasi Setelah Dijalankan.....</i>	<i>7</i>
Gambar 4. Screenshot Hasil Jawaban Soal 1 XML	14
Gambar 5. Screenshot Hasil Jawaban Soal 1 XML	15
Gambar 6 Screenshot Hasil Jawaban Soal 1 XML	16
Gambar 7 Screenshot Hasil Jawaban Soal 1	24
Gambar 8 Screenshot Hasil Jawaban Soal 1	25
Gambar 9 Screenshot Hasil Jawaban Soal 1	25

DAFTAR TABEL

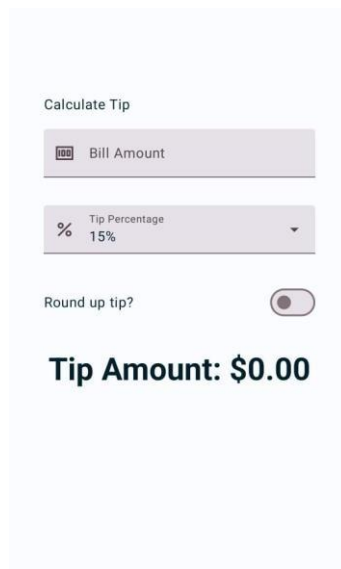
Tabel 1 Source Code Jawaban soal 1 XML	7
Tabel 2 Source Code Jawaban soal 1 XML	10
Tabel 3 Source Code Jawaban soal 1 XML	12
Tabel 4 Source Code Jawaban soal 1 XML	13
Tabel 5 Source Code Jawaban soal 1 Compose	20

SOAL 1

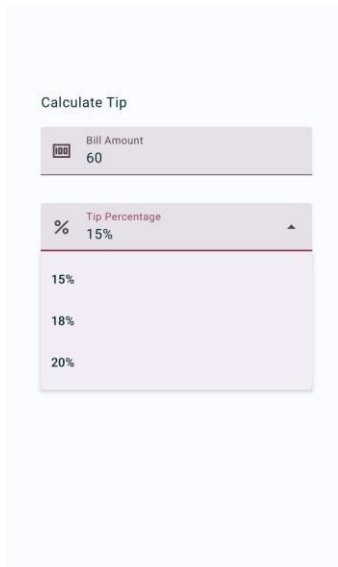
Soal Praktikum:

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:

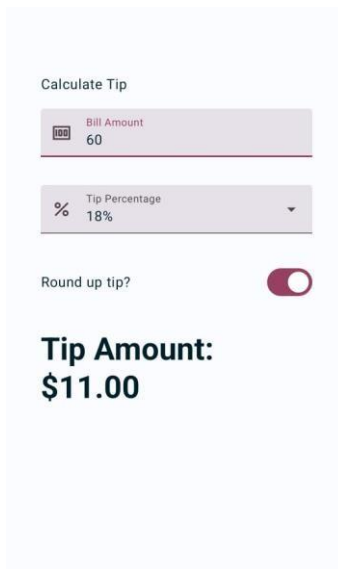
- Input biaya layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
- Pilihan persentase tip: Pengguna dapat memilih persentase tip yang diinginkan.
- Pengaturan pembulatan tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
- Tampilan hasil: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



Gambar 1 Tampilan Awal Aplikasi



Gambar 2 Tampilan Pilihan Persentase Tip



Gambar 3 Tampilan Aplikasi Setelah Dijalankan

A. Source Code XML

MainActivity.kt

Tabel 1 Source Code Jawaban soal 1 XML

1	package com.example.calculatortipx
2	
3	import android.os.Bundle

```

4 import android.text.Editable
5 import android.text.TextWatcher
6 import android.view.View
7 import android.view.ViewGroup
8 import android.widget.AdapterView
9 import android.widget.AdapterView
10 import android.widget.AdapterView
11 import androidx.appcompat.app.AppCompatActivity
12 import com.example.calculatorTipx.databinding.ActivityMainBinding
13 import java.text.NumberFormat
14 import java.util.*
15 import kotlin.math.ceil
16
17
18 class MainActivity : AppCompatActivity() {
19
20     private lateinit var binding: ActivityMainBinding
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         binding = ActivityMainBinding.inflate(layoutInflater)
25         setContentView(binding.root)
26
27         val tipOptions =
28 resources.getStringArray(R.array.tip_options)
29
30         val adapter = object : ArrayAdapter<String>(
31             this,
32             R.layout.spinner_item_with_icon,
33             R.id.text1,
34             tipOptions
35         ) {
36             override fun getView(position: Int, convertView: View?,
37 parent: ViewGroup): View {
38                 val view =
39 layoutInflater.inflate(R.layout.spinner_item_with_icon, parent,
40 false)
41                 val label = view.findViewById<TextView>(R.id.label)
42                 val text = view.findViewById<TextView>(R.id.text1)
43                 label.text = "Tip Percentage"
44                 text.text = tipOptions[position]
45                 return view
46             }
47
48             override fun getDropDownView(position: Int, convertView:
49 View?, parent: ViewGroup): View {
50                 val view =
51 layoutInflater.inflate(R.layout.spinner_dropdown_item_with_icon,
52 parent, false)
53                 val text = view.findViewById<TextView>(R.id.text1)
54                 text.text = tipOptions[position]
55                 return view

```



```

56         }
57     }
58     binding.tipOptionsSpinner.adapter = adapter
59     binding.tipOptionsSpinner.setSelection(0)
60     calculateTip() // panggil saat awal agar 15% dihitung
61     binding.serviceCostInput.addTextChangedListener(tipWatcher)
62     binding.tipOptionsSpinner.onItemSelectedListener =
63 spinnerListener
64     binding.roundUpSwitch.setOnCheckedChangeListener { _, _ ->
65 calculateTip() }
66     }
67
68     private val tipWatcher = object : TextWatcher {
69         override fun afterTextChanged(s: Editable?) = calculateTip()
70         override fun beforeTextChanged(s: CharSequence?, start: Int,
71 count: Int, after: Int) {}
72         override fun onTextChanged(s: CharSequence?, start: Int,
73 before: Int, count: Int) {}
74     }
75
76     private val spinnerListener = object :
77 AdapterView.OnItemSelectedListener {
78         override fun onItemSelected(parent: AdapterView<*>?, view:
79 View?, position: Int, id: Long) {
80             calculateTip()
81         }
82         override fun onNothingSelected(parent: AdapterView<*>?) {}
83     }
84
85     private fun calculateTip() {
86         val cost =
87 binding.serviceCostInput.text.toString().toDoubleOrNull() ?: 0.0
88
89         val tipPercent =
90 binding.tipOptionsSpinner.selectedItem.toString().removeSuffix("%").t
91 oDoubleOrNull() ?: 0.0
92         var tip = cost * tipPercent / 100
93         if (binding.roundUpSwitch.isChecked) {
94             tip = ceil(tip)
95         }
96         val formattedTip =
97 NumberFormat.getCurrencyInstance(Locale.US).format(tip)
98         binding.tipResult.text = "Tip Amount: $formattedTip"
99     }
100 }

```

activity_main.xml

Dalam XML ada beberapa file tambahan agar sama tampilannya dengan di gambar.

Tabel 2 Source Code Jawaban soal 1 XML

1	package com.example.calculatortipx
2	
3	import android.os.Bundle
4	import android.text.Editable
5	import android.text.TextWatcher
6	import android.view.View
7	import android.view.ViewGroup
8	import android.widget.AdapterView
9	import android.widget.ArrayAdapter
10	import android.widget.TextView
11	import androidx.appcompat.app.AppCompatActivity
12	import com.example.calculatortipx.databinding.ActivityMainBinding
13	import java.text.NumberFormat
14	import java.util.*
15	import kotlin.math.ceil
16	
17	
18	class MainActivity : AppCompatActivity() {
19	
20	private lateinit var binding: ActivityMainBinding
21	
22	override fun onCreate(savedInstanceState: Bundle?) {
23	super.onCreate(savedInstanceState)
24	binding = ActivityMainBinding.inflate(layoutInflater)
25	setContentView(binding.root)
26	
27	val tipOptions =
28	resources.getStringArray(R.array.tip_options)
29	
30	val adapter = object : ArrayAdapter<String>(
31	this,
32	R.layout.spinner_item_with_icon,
33	R.id.text1,
34	tipOptions
35) {
36	override fun getView(position: Int, convertView:
37	View?, parent: ViewGroup): View {
38	val view =
39	layoutInflater.inflate(R.layout.spinner_item_with_icon, parent,
40	false)
41	val label =
42	view.findViewById<TextView>(R.id.label)
43	val text = view.findViewById<TextView>(R.id.text1)
44	label.text = "Tip Percentage"
45	text.text = tipOptions[position]
46	return view
47	}
48	

```

49         override fun getDropDownView(position: Int,
50 convertView: View?, parent: ViewGroup): View {
51             val view =
52 layoutInflater.inflate(R.layout.spinner_dropdown_item_with_icon,
53 parent, false)
54             val text = view.findViewById<TextView>(R.id.text1)
55             text.text = tipOptions[position]
56             return view
57         }
58     }
59     binding.tipOptionsSpinner.adapter = adapter
60     binding.tipOptionsSpinner.setSelection(0)
61     calculateTip() // panggil saat awal agar 15% dihitung
62
63     binding.serviceCostInput.addTextChangedListener(tipWatcher)
64     binding.tipOptionsSpinner.onItemSelectedListener =
65     spinnerListener
66     binding.roundUpSwitch.setOnCheckedChangeListener { _, _ ->
67     calculateTip() }
68 }
69
70     private val tipWatcher = object : TextWatcher {
71         override fun afterTextChanged(s: Editable?) =
72         calculateTip()
73         override fun beforeTextChanged(s: CharSequence?, start:
74 Int, count: Int, after: Int) {}
75         override fun onTextChanged(s: CharSequence?, start: Int,
76 before: Int, count: Int) {}
77     }
78
79     private val spinnerListener = object :
80 AdapterView.OnItemSelectedListener {
81         override fun onItemSelected(parent: AdapterView<*>?, view:
82 View?, position: Int, id: Long) {
83             calculateTip()
84         }
85         override fun onNothingSelected(parent: AdapterView<*>?) {}
86     }
87
88     private fun calculateTip() {
89         val cost =
90 binding.serviceCostInput.text.toString().toDoubleOrNull() ?: 0.0
91
92         val tipPercent =
93 binding.tipOptionsSpinner.selectedItem.toString().removeSuffix("%"
94 ).toDoubleOrNull() ?: 0.0
95         var tip = cost * tipPercent / 100
96         if (binding.roundUpSwitch.isChecked) {
97             tip = ceil(tip)
98         }
99         val formattedTip =
100 NumberFormat.getCurrencyInstance(Locale.US).format(tip)

```

101	binding.tipResult.text = "Tip Amount: \$formattedTip"
102	}
103	}

spinner_item_with_icon.xml:

Tabel 3 Source Code Jawaban soal 1 XML

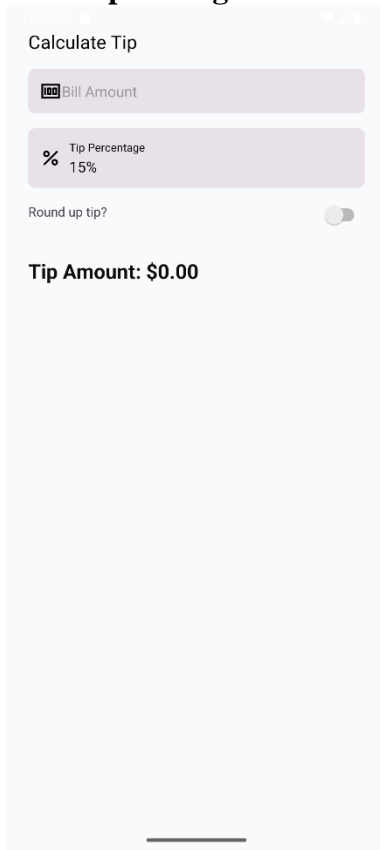
1	<LinearLayout
2	xmlns:android="http://schemas.android.com/apk/res/android"
3	android:layout_width="match_parent"
4	android:layout_height="wrap_content"
5	android:orientation="horizontal"
6	android:padding="12dp"
7	android:background="@drawable/edittext_background"
8	android:gravity="center_vertical">
9	
10	<ImageView
11	android:layout_width="24dp"
12	android:layout_height="24dp"
13	android:src="@drawable/percent"
14	android:layout_marginEnd="8dp" />
15	
16	<LinearLayout
17	android:layout_width="wrap_content"
18	android:layout_height="wrap_content"
19	android:orientation="vertical">
20	
21	<TextView
22	android:id="@+id/label"
23	android:layout_width="wrap_content"
24	android:layout_height="wrap_content"
25	android:text="Tip Percentage"
26	android:textColor="@color/primary_text"
27	android:textSize="12sp" />
28	
29	<TextView
30	android:id="@+id/text1"
31	android:layout_width="wrap_content"
32	android:layout_height="wrap_content"
33	android:textColor="@color/primary_text"
34	android:textSize="16sp"
35	android:paddingTop="2dp" />
36	</LinearLayout>
37	</LinearLayout>

spinner_dropdown_item_with_icon.xml

Tabel 4 Source Code Jawaban soal 1 XML

1	<LinearLayout
2	xmlns:android="http://schemas.android.com/apk/res/android"
3	android:layout_width="match_parent"
4	android:layout_height="wrap_content"
5	android:orientation="horizontal"
6	android:padding="12dp"
7	android:gravity="center_vertical"
8	android:background="@drawable/edittext_background">
9	
10	<ImageView
11	android:layout_width="24dp"
12	android:layout_height="24dp"
13	android:src="@drawable/percent"
14	android:layout_marginEnd="8dp" />
15	
16	<TextView
17	android:id="@+id/text1"
18	android:layout_width="wrap_content"
19	android:layout_height="wrap_content"
20	android:textColor="@color/primary_text"
21	android:textSize="16sp"/>
22	</LinearLayout>

B. Output Program



The screenshot shows a mobile application titled "Calculate Tip". It features two input fields: "Bill Amount" with a currency icon and "Tip Percentage" with a percentage icon, both set to "15%". Below these is a toggle switch for "Round up tip?". The output at the bottom is "Tip Amount: \$0.00".

Calculate Tip

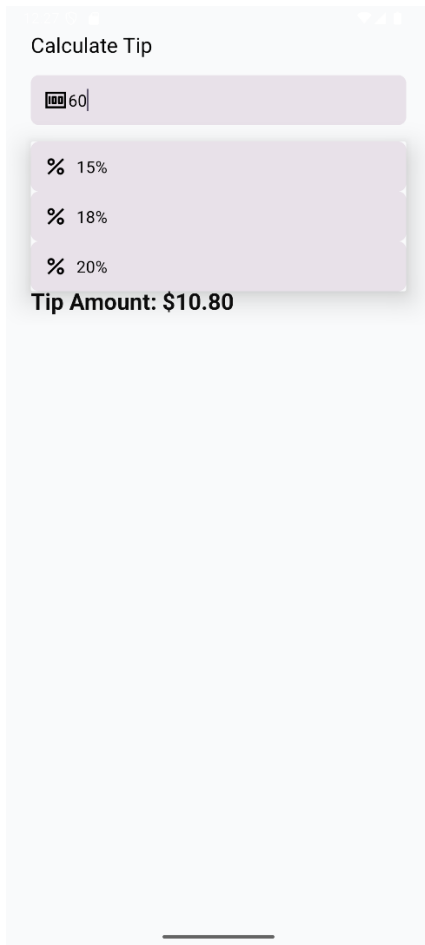
Bill Amount

Tip Percentage
15%

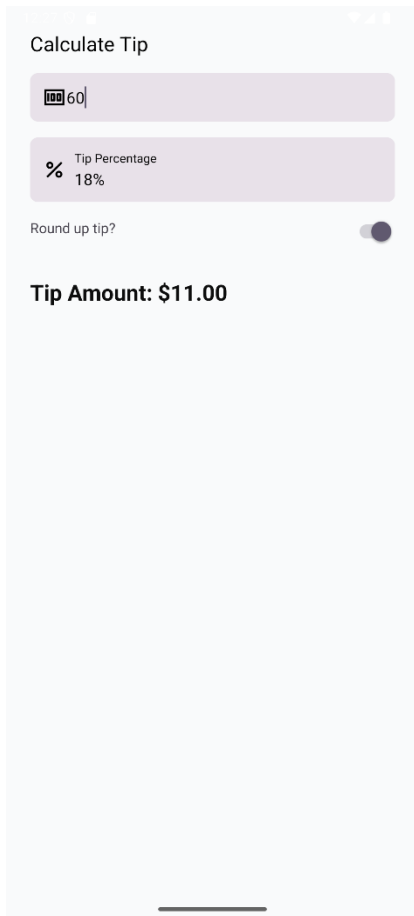
Round up tip?

Tip Amount: \$0.00

Gambar 4. Screenshot Hasil Jawaban Soal 1 XML



Gambar 5. Screenshot Hasil Jawaban Soal 1 XML



Gambar 6 Screenshot Hasil Jawaban Soal 1 XML

C. Pembahasan

MainActivity.kt:

Pada line 16–70, didefinisikan class MainActivity yang merupakan turunan dari AppCompatActivity, yaitu activity utama dalam aplikasi ini. Di dalamnya terdapat property binding yang digunakan untuk mengakses elemen-elemen UI dari layout *activity_main.xml*. Pada fungsi onCreate, pertama-tama binding diinisialisasi dengan `ActivityMainBinding.inflate` dan ditampilkan ke layar menggunakan `setContentView(binding.root)`. Kemudian, array `tipOptions` yang berisi daftar persentase tip (seperti 15%, 18%, 20%) diambil dari resource *strings.xml*. Setelah itu dibuatlah adapter kustom dari `ArrayAdapter`, menggunakan layout *spinner_item_with_icon* sebagai tampilan item spinner. Fungsi `getView` di dalam adapter digunakan untuk menampilkan layout spinner ketika tidak diklik, dengan label "Tip Percentage" dan teks persen yang diambil dari `tipOptions`. Sedangkan fungsi `getDropDownView` digunakan untuk menampilkan item *dropdown* ketika spinner dibuka, hanya menampilkan teks persen. Setelah adapter selesai dikonfigurasi, spinner `tipOptionsSpinner` dihubungkan dengan adapter, dan dipilih default ke indeks 0 agar langsung menampilkan nilai 15% pada awal aplikasi dijalankan. Fungsi `calculateTip()` langsung dipanggil untuk menghitung tip awal berdasarkan nilai

default. Selanjutnya, listener ditambahkan ke `EditText serviceCostInput` agar perhitungan tip bisa dilakukan setiap kali pengguna mengubah nominal biaya layanan. `Spinner` juga diberi listener untuk mendeteksi perubahan pilihan persentase tip. Terakhir, `switch roundUpSwitch` diberikan listener untuk menghitung ulang tip ketika pengguna mengaktifkan atau menonaktifkan pembulatan.

Pada line 47–52, didefinisikan objek `tipWatcher` sebagai implementasi `TextWatcher`. Objek ini digunakan untuk mendeteksi perubahan teks pada input biaya layanan. Hanya metode `afterTextChanged` yang diisi, yang akan memanggil fungsi `calculateTip()` setiap kali pengguna selesai mengetik.

Pada line 54–58, didefinisikan objek `spinnerListener` yang merupakan implementasi dari `AdapterView.OnItemSelectedListener`. Listener ini digunakan untuk menghitung ulang tip saat pengguna memilih persentase baru dari spinner. Fungsi `calculateTip()` dipanggil dalam `onItemSelected`, sedangkan fungsi `onNothingSelected` tidak melakukan apapun.

Pada line 60–70, didefinisikan fungsi `calculateTip()` sebagai inti logika perhitungan tip. Pertama, nominal biaya layanan dibaca dari `EditText` dan dikonversi ke `Double`. Jika input kosong atau tidak valid, maka akan dianggap sebagai 0. Kemudian persentase tip diambil dari item yang dipilih di spinner, dengan menghapus simbol persen "%" dan dikonversi menjadi nilai desimal. Nilai tip dihitung dengan mengalikan biaya dengan persen tip dan akan dibulatkan ke atas menggunakan fungsi `ceil` apabila `switch roundUpSwitch` diaktifkan. Hasil akhir dari tip diformat menggunakan `NumberFormat` dengan locale Amerika Serikat dan ditampilkan dalam bentuk teks pada `TextView` dengan format `Tip Amount: $[nominal]`.

activity_main.xml:

Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8. Selanjutnya pada line 2–5, digunakan tag `ConstraintLayout` dari Jetpack `ConstraintLayout` sebagai layout utama dengan namespace `Android (xmlns:android)` dan `App (xmlns:app)` yang digunakan untuk mengatur atribut-atribut khusus. Layout ini memiliki ukuran penuh terhadap parent (`match_parent`) baik untuk lebar maupun tinggi, ditambahkan padding sebesar 24dp di seluruh sisi, serta latar belakang diatur menggunakan warna `@color/background_light`.

Pada line 7–14, ditambahkan komponen `TextView` dengan id `titleText`, yang berfungsi sebagai judul tampilan dengan teks "Calculate Tip". Ukuran teks diatur sebesar 20sp dan warnanya menggunakan warna primer `@color/primary_text`. Komponen ini diposisikan di bagian atas dan kiri layout menggunakan `constraint layout_constraintTop_toTopOf="parent"` dan `layout_constraintStart_toStartOf="parent"`.

Pada line 16–28, didefinisikan komponen `EditText` dengan id `serviceCostInput` yang digunakan untuk menginput nominal tagihan atau biaya layanan. Lebar elemen diatur `0dp` agar mengikuti constraint start dan end (disebut `match constraints`), sementara tinggi mengikuti kontennya. `EditText` ini memiliki hint "Bill Amount", tipe input `numberDecimal` agar hanya menerima angka desimal, serta icon di sisi kiri yang berasal dari `drawable money`. Selain itu, terdapat padding internal sebesar `12dp`, padding antara icon dan teks sebesar `8dp`, latar belakang khusus yang ditentukan oleh `@drawable/edittext_background`, dan warna teks menggunakan `@color/primary_text`. Posisi `EditText` ini diatur tepat di bawah `titleText` dengan margin atas sebesar `16dp` dan direntangkan dari kiri ke kanan layout.

Pada line 30–38, didefinisikan komponen `Spinner` dengan id `tipOptionsSpinner` yang digunakan sebagai dropdown untuk memilih persentase tip. Lebar nya juga diatur `0dp` agar mengikuti constraint kiri dan kanan, dan tinggi menyesuaikan isi. `Spinner` ini memiliki margin atas `16dp` dari `serviceCostInput`, latar belakang dari `drawable kustom edittext_background`, serta tinggi minimum `48dp` untuk menjaga konsistensi ukuran elemen.

Pada line 40–46, ditambahkan `TextView` dengan id `roundUpLabel` yang menampilkan teks "Round up tip?" untuk menanyakan apakah hasil tip perlu dibulatkan. Ukuran teks diatur sebesar `20sp`, dan posisinya diatur tepat di bawah `Spinner tipOptionsSpinner`, dengan margin atas `16dp` dan disejajarkan ke kiri layout.

Pada line 48–54, didefinisikan komponen `Switch` dengan id `roundUpSwitch` yang digunakan pengguna untuk mengaktifkan atau menonaktifkan fitur pembulatan tip. `Switch` ini memiliki margin atas `16dp`, tinggi dan lebar minimum masing-masing `48dp` untuk memastikannya cukup besar untuk disentuh, dan posisinya disejajarkan ke kanan layout, tepat di bawah `tipOptionsSpinner`.

Pada line 56–63, didefinisikan `TextView` dengan id `tipResult` yang digunakan untuk menampilkan hasil akhir perhitungan tip. Teks default-nya adalah "Tip Amount: \$0.00", dengan ukuran teks sebesar `22sp` dan gaya teks tebal. Warna teksnya diatur menggunakan `@color/tip_result_text`. `TextView` ini diposisikan di bawah `Switch roundUpSwitch` dengan margin atas `32dp` dan disejajarkan ke kiri layout.

spinner_item_with_icon.xml:

Pada line 1–8, digunakan elemen `LinearLayout` sebagai root layout dengan orientasi horizontal. Layout ini diatur agar memiliki lebar penuh (`match_parent`) dan tinggi menyesuaikan isi (`wrap_content`). Padding sebesar `12dp` ditambahkan ke seluruh sisi layout untuk memberi ruang di dalam elemen, dan latar belakangnya menggunakan `drawable kustom @drawable/edittext_background`, yang kemungkinan memberikan border atau efek khusus. Atribut `gravity="center_vertical"` digunakan agar konten di dalam layout disejajarkan secara vertikal di tengah.

Pada line 10–13, ditambahkan komponen `ImageView` yang menampilkan ikon bergambar persentase (`@drawable/percent`) dengan ukuran 24x24dp. Margin end sebesar 8dp ditambahkan agar ada jarak antara gambar dan elemen di sebelah kanannya.

Pada line 15–22, terdapat `LinearLayout` kedua di dalam layout utama, dengan orientasi vertikal. Layout ini digunakan untuk menumpuk dua `TextView` secara vertikal, yaitu label dan nilai persen tip. Lebar dan tinggi layout ini disesuaikan dengan ukuran kontennya (`wrap_content`).

Pada line 24–29, dideklarasikan `TextView` pertama dengan id `label`, yang berfungsi menampilkan label teks "Tip Percentage". Ukuran teks diatur sebesar 12sp dan warna teks menggunakan warna `@color/primary_text`.

Pada line 31–36, didefinisikan `TextView` kedua dengan id `text1` yang berfungsi menampilkan nilai persen tip yang dipilih, seperti "15%", "18%", atau "20%". Ukuran teksnya sedikit lebih besar, yaitu 16sp, dan warna teksnya sama seperti label. Padding atas sebesar 2dp ditambahkan agar ada jarak antara label dan nilai teks ini.

spinner_dropdown_item_with_icon.xml:

Pada line 1–8, digunakan elemen `LinearLayout` sebagai layout utama dengan orientasi horizontal, yang berarti elemen-elemen di dalamnya akan ditata secara mendatar dari kiri ke kanan. Layout ini memiliki lebar penuh (`match_parent`) dan tinggi yang menyesuaikan isi kontennya (`wrap_content`). Padding sebesar 12dp ditambahkan ke seluruh sisi untuk memberi ruang di dalam layout, dan atribut `gravity="center_vertical"` digunakan agar konten disejajarkan secara vertikal di tengah. Selain itu, latar belakang layout menggunakan `drawable @drawable/edittext_background`, kemungkinan besar untuk memberikan efek visual seperti border atau bayangan agar terlihat seperti input field.

Pada line 10–13, dideklarasikan sebuah `ImageView` yang digunakan untuk menampilkan ikon bergambar persentase (`@drawable/percent`). Ukuran ikon ditetapkan sebesar 24x24dp dan diberi `layout_marginEnd` sebesar 8dp untuk memberikan jarak antara ikon dan elemen di sebelah kanannya, sehingga tata letaknya tetap rapi.

Pada line 15–20, terdapat `TextView` dengan id `text1` yang digunakan untuk menampilkan nilai teks dari opsi persentase tip yang dipilih, seperti "15%", "18%", atau "20%". Ukuran teks ditetapkan sebesar 16sp agar mudah terbaca, dan warna teks ditentukan dengan menggunakan warna `@color/primary_text`, agar selaras dengan tema aplikasi.

D. Source Code Compose MainActivity.kt

Tabel 5 Source Code Jawaban soal 1 Compose

```
1 package com.example.calculatortipj
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import androidx.activity.enableEdgeToEdge
7 import androidx.compose.foundation.layout.Arrangement
8 import androidx.compose.foundation.layout.Column
9 import androidx.compose.foundation.layout.Spacer
10 import androidx.compose.foundation.layout.fillMaxWidth
11 import androidx.compose.foundation.layout.height
12 import androidx.compose.foundation.layout.padding
13 import androidx.compose.foundation.rememberScrollState
14 import androidx.compose.foundation.text.KeyboardOptions
15 import androidx.compose.foundation.verticalScroll
16 import androidx.compose.material3.MaterialTheme
17 import androidx.compose.material3.Text
18 import androidx.compose.material3.TextField
19 import androidx.compose.runtime.Composable
20 import androidx.compose.runtime.getValue
21 import androidx.compose.runtime.mutableStateOf
22 import androidx.compose.runtime.saveable.rememberSaveable
23 import androidx.compose.runtime.setValue
24 import androidx.compose.ui.Alignment
25 import androidx.compose.ui.Modifier
26 import androidx.compose.ui.res.painterResource
27 import androidx.compose.ui.text.input.KeyboardType
28 import androidx.compose.ui.tooling.preview.Preview
29 import androidx.compose.ui.unit.dp
30 import com.example.calculatortipj.ui.theme.CalculatorTipJTheme
31 import androidx.compose.material3.Icon
32 import androidx.compose.material3.Switch
33 import androidx.compose.foundation.layout.Row
34 import androidx.compose.ui.text.input.ImeAction
35 import java.text.NumberFormat
36 import androidx.compose.material3.*
37 import androidx.compose.material3.ExposedDropdownMenuBox
38 import androidx.compose.material3.ExposedDropdownMenuDefaults
39 import androidx.compose.runtime.remember
40
41
42 class MainActivity : ComponentActivity() {
43     override fun onCreate(savedInstanceState: Bundle?) {
44         super.onCreate(savedInstanceState)
45         enableEdgeToEdge()
46         setContent {
```

```

47         CalculatorTipJTheme {
48             CalculatorTipApp()
49         }
50     }
51 }
52 }
53
54 @Composable
55 fun CalculatorTipApp() {
56     var amountInput by rememberSaveable { mutableStateOf("") }
57     var selectedTipPercent by rememberSaveable {
58         mutableStateOf("15%") }
59     var roundUp by rememberSaveable { mutableStateOf(false) }
60
61     val amount = amountInput.toDoubleOrNull() ?: 0.0
62     val tipPercent =
63         selectedTipPercent.removeSuffix("%").toDoubleOrNull() ?: 0.0
64     val tip = calculateTip(amount, tipPercent, roundUp)
65
66     Column(
67         modifier = Modifier
68             .verticalScroll(rememberScrollState())
69             .padding(40.dp),
70         horizontalAlignment = Alignment.CenterHorizontally,
71         verticalArrangement = Arrangement.Center
72     ) {
73         Text(
74             text = "Calculator Tip",
75             modifier = Modifier
76                 .padding(bottom = 16.dp, top = 16.dp)
77                 .align(alignment = Alignment.Start)
78         )
79         EditNumberField(
80             label = "Bill Amount",
81             value = amountInput,
82             leadingIcon = R.drawable.money,
83             onValueChange = { amountInput = it },
84             keyboardOptions = KeyboardOptions(
85                 keyboardType = KeyboardType.Number,
86                 imeAction = ImeAction.Next
87             ),
88             modifier = Modifier
89                 .padding(bottom = 32.dp)
90                 .fillMaxWidth()
91         )
92         TipPercentageDropdown(
93             selectedOption = selectedTipPercent,
94             onOptionSelected = { selectedTipPercent = it },
95             modifier = Modifier
96                 .padding(bottom = 32.dp)
97                 .fillMaxWidth()
98         )

```

```

99         )
100         roundTheTipRow(
101             roundUp = roundUp,
102             onCheckedChange = { roundUp = it },
103             modifier = Modifier
104                 .padding(bottom = 32.dp)
105         )
106         Text(
107             text = "Tip Amount: $tip",
108             style = MaterialTheme.typography.displaySmall,
109         )
110         Spacer(modifier = Modifier.height(150.dp))
111     }
112 }
112
113 @Composable
114 fun EditNumberField(
115     label: String,
116     value: String,
117     leadingIcon: Int,
118     onValueChange: (String) -> Unit,
119     keyboardOptions: KeyboardOptions,
121     modifier: Modifier = Modifier
122 ) {
123     TextField(
124         value = value,
125         singleLine = true,
126         modifier = modifier,
127         onValueChange = onValueChange,
128         label = { Text(text = label) },
129         leadingIcon = { Icon(painterResource(id = leadingIcon),
130 contentDescription = null) },
131         keyboardOptions = keyboardOptions
132     )
133 }
134
135 @Composable
136 fun roundTheTipRow(
137     roundUp: Boolean,
138     onCheckedChange: (Boolean) -> Unit,
139     modifier: Modifier = Modifier
140 ) {
141     Row(
142         modifier = modifier
143             .fillMaxWidth()
144             .padding(16.dp),
145         verticalAlignment = Alignment.CenterVertically,
146         horizontalArrangement = Arrangement.SpaceBetween
147     ) {
148         Text(text = "Round up tip")
149         Switch(
150             checked = roundUp,

```

```

151         onCheckedChange = onCheckedChange
152     )
153 }
154 }
155
156 private fun calculateTip(amount: Double, tipPercent: Double,
157 roundUp: Boolean): String {
158     var tip = tipPercent / 100 * amount
159     if (roundUp) {
160         tip = kotlin.math.ceil(tip)
161     }
162     return
163     NumberFormat.getCurrencyInstance(java.util.Locale.US).format(tip)
164 }
165
166 @OptIn(ExperimentalMaterial3Api::class)
167 @Composable
168 fun TipPercentageDropdown(
169     selectedOption: String,
170     onOptionSelected: (String) -> Unit,
171     modifier: Modifier = Modifier
172 ) {
173     val options = listOf("15%", "18%", "20%")
174     var expanded by remember { mutableStateOf(false) }
175
176     ExposedDropdownMenuBox(
177         expanded = expanded,
178         onExpandedChange = { expanded = !expanded },
179         modifier = modifier
180     ) {
181         TextField(
182             value = selectedOption,
183             onValueChange = {},
184             readOnly = true,
185             label = { Text("Tip Percentage") },
186             trailingIcon = {
187                 ExposedDropdownMenuDefaults.TrailingIcon(expanded
188 = expanded)
189             },
190             modifier = Modifier.menuAnchor().fillMaxWidth()
191         )
192
193         ExposedDropdownMenu(
194             expanded = expanded,
195             onDismissRequest = { expanded = false }
196         ) {
197             options.forEach { option ->
198                 DropdownMenuItem(
199                     text = { Text(option) },
200                     onClick = {
201                         onOptionSelected(option)
202                         expanded = false

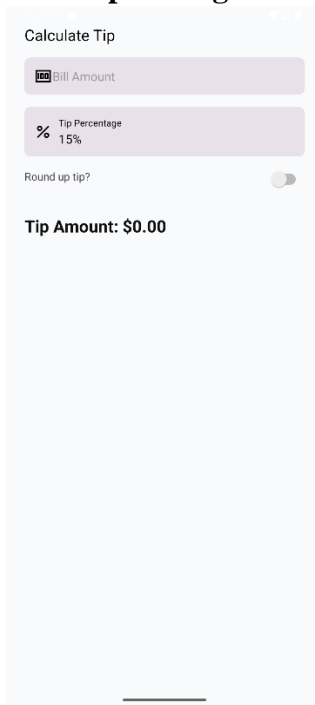
```

```

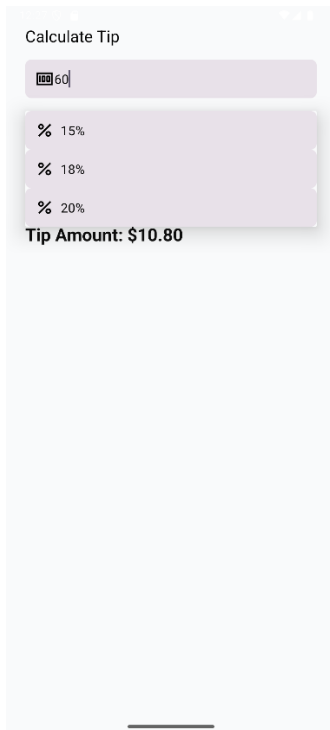
203         }
204     )
205 }
206 }
207 }
208 }
209
210 @Preview(showBackground = true)
211 @Composable
212 fun DefaultPreview() {
213     CalculatorTipJTheme {
214         CalculatorTipApp()
215     }
216 }

```

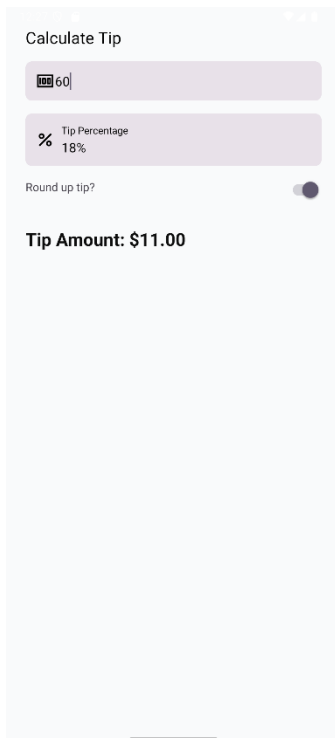
E. Output Program



Gambar 7 Screenshot Hasil Jawaban Soal 1



Gambar 8 Screenshot Hasil Jawaban Soal 1



Gambar 9 Screenshot Hasil Jawaban Soal 1

F. Pembahasan MainActivity.kt:

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.calculatortipj`.

Pada line 3–39, dilakukan import terhadap berbagai komponen dan library Jetpack Compose seperti `Column`, `TextField`, `Text`, `Modifier`, dan `MaterialTheme`, serta beberapa komponen tambahan seperti `Switch`, `Icon`, dan `KeyboardOptions` yang dibutuhkan untuk membangun UI aplikasi menggunakan Jetpack Compose, bukan XML.

Pada line 54-112, didefinisikan class `MainActivity` yang merupakan turunan dari `ComponentActivity`, yaitu activity dasar untuk aplikasi yang menggunakan Jetpack Compose. Di dalam fungsi `onCreate`, dipanggil fungsi `enableEdgeToEdge()` agar aplikasi tampil full screen. Lalu, `setContent` digunakan untuk mengatur konten UI dari aplikasi, dan fungsi `CalculatorTipApp()` dipanggil sebagai entry point untuk komponen Compose utama. Kemudian didefinisikan fungsi `CalculatorTipApp()` sebagai komponen utama UI aplikasi. Di dalamnya dideklarasikan beberapa state seperti `amountInput`, `selectedTipPercent`, dan `roundUp` dengan `rememberSaveable`, agar state tetap terjaga saat konfigurasi berubah. Kemudian dihitung jumlah tip menggunakan fungsi `calculateTip()`. UI ditampilkan dalam `Column` dengan pengaturan scroll dan padding.

Pada baris 113-133 Komponen yang ditampilkan di dalamnya antara lain: judul aplikasi dengan `Text`, input nominal tagihan dengan `EditNumberField`, dropdown pilihan persen tip dengan `TipPercentageDropdown`, toggle switch pembulatan tip dengan `roundTheTipRow`, serta hasil tip yang dihitung. Kemudian didefinisikan fungsi `EditNumberField()` untuk membuat input angka nominal tagihan. Fungsi ini menampilkan `TextField` dengan icon di bagian depan menggunakan `leadingIcon`, label input, dan opsi keyboard khusus angka. Fungsi ini dipanggil di `CalculatorTipApp()`.

Pada baris ke 135-208 itu mendefinisikan fungsi `roundTheTipRow()` untuk menampilkan teks dan komponen `Switch` (saklar) yang dapat mengatur apakah jumlah tip dibulatkan atau tidak. Komponen ini diletakkan dalam `Row` dengan padding dan alignment horizontal yang sesuai. Lalu mendefinisikan fungsi `calculateTip()` untuk menghitung jumlah tip berdasarkan input pengguna. Perhitungan dilakukan dengan mengalikan nominal tagihan dengan persen tip. Jika `roundUp` bernilai true, maka hasil tip dibulatkan ke atas menggunakan `ceil`. Hasil akhirnya diformat menjadi format mata uang lokal Amerika Serikat. Kemudian mendefinisikan fungsi `DefaultPreview()` sebagai fungsi preview bawaan dari Jetpack Compose untuk menampilkan tampilan UI dalam Android Studio saat sedang dikembangkan, tanpa perlu menjalankan aplikasi di emulator atau perangkat. Kemudian Mendefinisikan fungsi `TipPercentageDropdown()` sebagai komponen dropdown menu untuk memilih persen tip (15%, 18%, 20%). Digunakan `ExposedDropdownMenuBox` dan `ExposedDropdownMenu` dari Material 3. `State expanded` digunakan untuk menampilkan atau menyembunyikan dropdown, dan

komponen `TextField` digunakan untuk menampilkan nilai yang dipilih. Ketika pengguna memilih salah satu opsi, dropdown akan menutup secara otomatis dan nilai terpilih akan dikirim ke parameter `onOptionSelected`.

Soal 2

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

XML merupakan metode atau cara tradisional atau imperatif yang telah digunakan untuk membuat UI, metode ini bersifat imperatif, untuk elemen elemen UI dituliskan dalam file terpisah dengan file nama XML, kemudian dihubungkan dengan file activity atau fragment. Kelebihan menggunakan XML antara lain adalah kestabilannya karena metode ini sudah matang dan luas digunakan, dokumentasi yang melimpah, dan kemudahan dalam memisahkan logika dan tampilan, tapi ada beberapa kekurangan untuk XML seperti struktur code yang sangat susah dan ribet, sering sekali struktur kodenya panjang-panjang, dalam menangani UI yang dinamis kurang fleksibel dan performanya seoptimal metode baru yaitu Jetpack Compose.

Jetpack Compose merupakan metode modern dari google berbasis bahasa kotlin, Compose bersifat deklaratif karena tidak dipisah seperti XML, kelebihannya lebih fleksibel dan ringkas, serta secara alami mendukung UI bersifat reaktif, sehingga sangat cocok digunakan pada aplikasi yang memerlukan perubahan tampilan jika ada perubahan data. Compose juga dengan baik dengan fitur fitur modern. Kelebihan lainnya adalah kemudahan dalam membentuk komponen UI custom tanpa memerlukan banyak kode tambahan meski sangat fleksibel cepat dan ringkas, Jetpack Compose baru dan beberapa fiturnya belum stabil.

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/Easydaf/Praktikum_Mobile/tree/main/Modul2