

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2**



Android Layout With Compose

Oleh:

Muhammad Adh-Dhiya'Us Salim

NIM. 2310817210022

**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 Praktikum ini dikerjakan oleh:

Nama Praktikan : Muhammad Adh-Dhiya'Us Salim
NIM : 2310817210022

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Raka Azwar
NIM. 2210817210012

Ir. Eka Setya Wijaya, S.T., M.Kom.
NIP. 198205082008011010

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI.....	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL.....	5
SOAL 1	6
A. Source Code	7
B. Output Program.....	14
C. Pembahasan.....	17
D. Tautan Git	21
SOAL 2	22
A. Jawaban.....	22

DAFTAR GAMBAR

Gambar 1 Tampilan Awal Aplikasi	6
Gambar 2 Tampilan Pilihan Persentasi Tip	6
Gambar 3 Tampilan Aplikasi Setelah Dijalanmkan	7
Gambar 4 Screenshot Hasil Jawaban Soal 1 - Compose	14
Gambar 5 Screenshot Hasil Jawaban Soal 1 - Compose	14
Gambar 6 Screenshot Hasil Jawaban Soal 1 - Compose	15
Gambar 7 Screenshot Hasil Jawaban Soal 1 – Compose.....	15
Gambar 8 Screenshot Hasil Jawaban Soal 1 - XML.....	16
Gambar 9 Screenshot Hasil Jawaban Soal 1 - XML.....	16
Gambar 10 Screenshot Hasil Jawaban Soal 1 - XML.....	17
Gambar 11 Screenshot Hasil Jawaban Soal 1 - XML.....	17

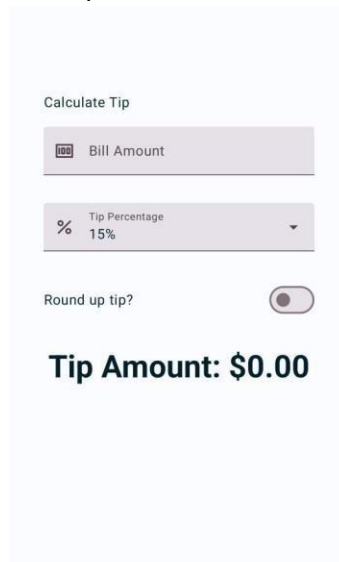
DAFTAR TABEL

Tabel 1 Soruce Code Soal 1 - Compose	10
Tabel 2 Soruce Code Soal 1 – XML	12
Tabel 3 Soruce Code Soal 1 - XML.....	13

SOAL 1

Soal Praktikum:

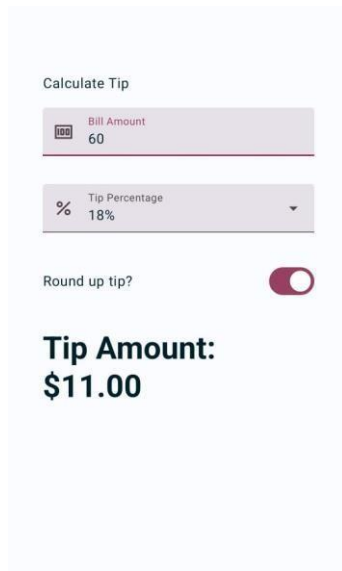
1. Buatlah sebuah aplikasi kalkulator tip menggunakan XML dan Jetpack Compose yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:
 - a. Input biaya layanan: Pengguna dapat memasukkan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



Gambar 1 Tampilan Awal Aplikasi



Gambar 2 Tampilan Pilihan Persentase Tip



Gambar 3 Tampilan Aplikasi Setelah Dijalankan

A. Source Code

MainActivity.kt / Compose

```

1 package com.example.kalkulator
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import androidx.compose.foundation.layout.*
7 import androidx.compose.foundation.text.KeyboardOptions
8 import androidx.compose.material.icons.Icons
9 import androidx.compose.material.icons.filled.Percent
10 import androidx.compose.material.icons.filled.Receipt
11 import androidx.compose.material3.*
12 import androidx.compose.runtime.*
13 import androidx.compose.ui.Alignment
14 import androidx.compose.ui.Modifier
15 import androidx.compose.ui.text.font.FontWeight
16 import androidx.compose.ui.text.input.KeyboardType
17 import androidx.compose.ui.tooling.preview.Preview
18 import androidx.compose.ui.unit.dp
19 import androidx.compose.ui.unit.sp
20 import com.example.kalkulator.ui.theme.KalkulatorTheme
21 import java.text.NumberFormat
22
23 class MainActivity : ComponentActivity() {
24     override fun onCreate(savedInstanceState: Bundle?) {
25         super.onCreate(savedInstanceState)
26         setContent {
27             KalkulatorTheme {
28                 Surface(
29                     modifier = Modifier.fillMaxSize(),
30                     color =
31 MaterialTheme.colorScheme.background
32 )

```

```

33         TipCalculatorScreen()
34     }
35 }
36 }
37 }
38 }
39
40 @OptIn(ExperimentalMaterial3Api::class)
41 @Composable
42 fun TipCalculatorScreen() {
43     var billAmountInput by remember { mutableStateOf("") }
44     var roundUp by remember { mutableStateOf(false) }
45
46     val tipOptions = listOf("15%", "18%", "20%")
47     var expanded by remember { mutableStateOf(false) }
48     var selectedTipOption by remember {
49         mutableStateOf(tipOptions[0])
50     }
51
52     val amount = billAmountInput.toDoubleOrNull() ?: 0.0
53     val tipPercent = selectedTipOption.removeSuffix("%").toDoubleOrNull() ?: 0.0
54     val tip = calculateTip(amount, tipPercent, roundUp)
55
56     Column(
57         modifier = Modifier
58             .padding(32.dp)
59             .fillMaxWidth(),
60         horizontalAlignment = Alignment.CenterHorizontally,
61         verticalArrangement = Arrangement.spacedBy(16.dp)
62     ) {
63         Text(
64             text = "Calculate Tip",
65             fontSize = 24.sp,
66             modifier = Modifier.align(Alignment.Start)
67         )
68
69         EditNumberField(
70             label = "Bill Amount",
71             value = billAmountInput,
72             onValueChange = { billAmountInput = it },
73             leadingIcon = { Icon(Icons.Filled.Receipt,
74 contentDescription = "Bill Amount") }
75         )
76
77         ExposedDropDownMenuBox(
78             expanded = expanded,
79             onExpandedChange = { expanded = !expanded },
80         ) {
81             OutlinedTextField(
82                 value = selectedTipOption,
83                 onValueChange = {},
84                 readOnly = true,
85                 label = { Text("Tip Percentage") },
86                 // MODIFIKASI: Tambahkan leadingIcon di sini
87                 leadingIcon = { Icon(Icons.Filled.Percent,
88 contentDescription = "Tip Percentage") },

```



```

89         trailingIcon = {
90
91     ExposedDropDownMenuDefaults.TrailingIcon(expanded = expanded)
92     },
93     modifier = Modifier
94         .menuAnchor()
95         .fillMaxWidth()
96     )
97     ExposedDropDownMenu(
98         expanded = expanded,
99         onDismissRequest = { expanded = false }
100     ) {
101         tipOptions.forEach { selectionOption ->
102             DropdownMenuItem(
103                 text = { Text(selectionOption) },
104                 onClick = {
105                     selectedTipOption =
106 selectionOption
107                     expanded = false
108                 }
109             )
110         }
111     }
112 }
113 }
114
115 RoundTheTipRow(
116     roundUp = roundUp,
117     onRoundUpChanged = { roundUp = it }
118 )
119
120 Spacer(modifier = Modifier.height(24.dp))
121
122 Text(
123     text = "Tip Amount: $tip",
124     fontSize = 28.sp,
125     fontWeight = FontWeight.Bold
126 )
127 }
128 }
129 }
130
131 @Composable
132 fun EditNumberField(
133     label: String,
134     value: String,
135     onValueChange: (String) -> Unit,
136     leadingIcon: @Composable () -> Unit
137 ) {
138     OutlinedTextField(
139         value = value,
140         onValueChange = onValueChange,
141         label = { Text(label) },
142         leadingIcon = leadingIcon,
143         modifier = Modifier.fillMaxWidth(),
144         singleLine = true,
145         keyboardOptions = KeyboardOptions(keyboardType =
146 KeyboardType.Number)

```

147	<pre>) } @Composable fun RoundTheTipRow(roundUp: Boolean, onRoundUpChanged: (Boolean) -> Unit, modifier: Modifier = Modifier) { Row(modifier = modifier.fillMaxWidth().size(48.dp), verticalAlignment = Alignment.CenterVertically) { Text(text = "Round up tip?") Spacer(modifier = Modifier.weight(1f)) Switch(checked = roundUp, onCheckedChange = onRoundUpChanged) } } private fun calculateTip(amount: Double, tipPercent: Double = 15.0, roundUp: Boolean): String { var tip = tipPercent / 100 * amount if (roundUp) { tip = kotlin.math.ceil(tip) } return NumberFormat.getCurrencyInstance().format(tip) } @Preview(showBackground = true) @Composable fun DefaultPreview() { KalkulatorTheme { TipCalculatorScreen() } } </pre>
-----	---

Tabel 1 Source Code Soal 1 - Compose

activity_main.xml / XML

1	<?xml	version="1.0"	encoding="utf-8"?>
2	<LinearLayout		
3	xmlns:android="http://schemas.android.com/apk/res/android"		
4	xmlns:tools="http://schemas.android.com/tools"		
5	android:layout_width="match_parent"		
6	android:layout_height="match_parent"		
7	android:orientation="vertical"		
8	android:padding="32dp"		
9	tools:context=".MainActivity">		
10	<TextView		
11	android:layout_width="wrap_content"		
12	android:layout_height="wrap_content"		
13	android:text="Calculate		Tip"
14	android:textSize="24sp"		/>

```

1      <com.google.android.material.textfield.TextInputLayout
3          android:id="@+id/bill_amount_layout"
1          android:layout_width="match_parent"
4          android:layout_height="wrap_content"
1          android:layout_marginTop="16dp"
5          android:hint="Bill                                Amount">
1
6          <com.google.android.material.textfield.TextInputEditText
1              android:id="@+id/bill_amount_edit_text"
7              android:layout_width="match_parent"
1              android:layout_height="wrap_content"
8              android:inputType="numberDecimal"                                />
1
9      </com.google.android.material.textfield.TextInputLayout>
2
0      <com.google.android.material.textfield.TextInputLayout
2          android:id="@+id/tip_percentage_layout"
1
2          style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.
2          ExposedDropdownMenu"
2          android:layout_width="match_parent"
3          android:layout_height="wrap_content"
2          android:layout_marginTop="16dp"
4          android:hint="Tip                                Percentage">
2
5          <AutoCompleteTextView
2              android:id="@+id/tip_percentage_autocomplete"
6              android:layout_width="match_parent"
2              android:layout_height="wrap_content"
7              android:inputType="none"                                />
2
8      </com.google.android.material.textfield.TextInputLayout>
2      <com.google.android.material.switchmaterial.SwitchMaterial
9          android:id="@+id/round_up_switch"
3          android:layout_width="match_parent"
0          android:layout_height="wrap_content"
3          android:layout_marginTop="16dp"
1          android:minHeight="48dp"
3          android:text="Round                                up                                tip?"                                />
2
3      <TextView
3          android:id="@+id/tip_result_text_view"
3          android:layout_width="wrap_content"
4          android:layout_height="wrap_content"
3          android:layout_gravity="center_horizontal"
5          android:layout_marginTop="40dp"
3          android:text="Tip                                Amount:                                $0.00"
6          android:textSize="28sp"
3          android:textStyle="bold"                                />
7
3      </LinearLayout>
8
3
9
4
0

```

4	
1	
4	
2	
4	
3	

Tabel 2 Sorce Code Soal 1 – XML

MainActivity.kt / XML

```

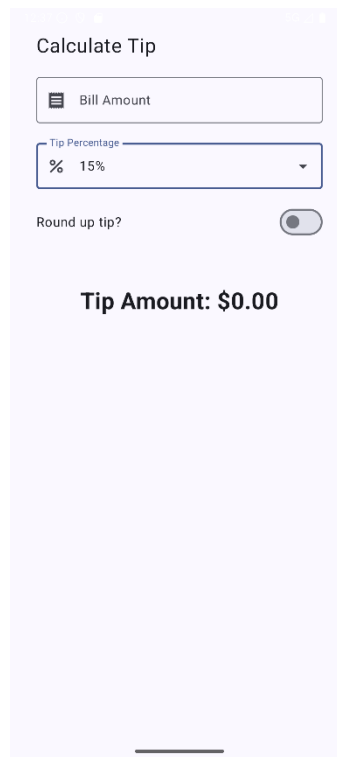
1 package com.example.tipxml
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.text.Editable
6 import android.text.TextWatcher
7 import android.widget.ArrayAdapter
8 import com.example.tipxml.databinding.ActivityMainBinding
9 import java.text.NumberFormat
10
11 class MainActivity : AppCompatActivity() {
12
13     private lateinit var binding: ActivityMainBinding
14
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         binding = ActivityMainBinding.inflate(layoutInflater)
18         setContentView(binding.root)
19
20         val tipOptions =
21 resources.getStringArray(R.array.tip_percentage_options)
22         val adapter = ArrayAdapter(this,
23 android.R.layout.simple_spinner_dropdown_item, tipOptions)
24         binding.tipPercentageAutocomplete.setAdapter(adapter)
25
26 binding.tipPercentageAutocomplete.setText(tipOptions[0], false)
27
28         setupListeners()
29         calculateAndDisplayTip()
30     }
31
32     private fun setupListeners() {
33         binding.billAmountEditText.addTextChangedListener {
34 calculateAndDisplayTip()

```

35	binding.roundUpSwitch.setOnCheckedChangeListener { _, _
36	-> calculateAndDisplayTip() }
37	
38	
39	binding.tipPercentageAutocomplete.setOnItemClickListener { _, _
40	-> calculateAndDisplayTip() }
41	
42	
43	
44	
45	private fun calculateAndDisplayTip() {
46	val billAmountString =
47	binding.billAmountEditText.text.toString()
48	val tipPercent =
49	binding.tipPercentageAutocomplete.text.toString()
	val amount = billAmountString.toDoubleOrNull() ?: 0.0
	val tipPercent =
	tipPercentString.removeSuffix("%").toDoubleOrNull() ?: 0.0
	val roundUp = binding.roundUpSwitch.isChecked
	var tip = tipPercent / 100 * amount
	if (roundUp) {
	tip = kotlin.math.ceil(tip)
	}
	val formattedTip =
	NumberFormat.getCurrencyInstance().format(tip)
	binding.tipResultTextView.text = "Tip Amount:
	\$formattedTip"
	}
	private fun
	android.widget.EditText.addTextChangedListener(onTextChanged:
	(String) -> Unit) {
	this.addTextChangedListener(object : TextWatcher {
	override fun beforeTextChanged(s: CharSequence?,
	start: Int, count: Int, after: Int) {}
	override fun onTextChanged(s: CharSequence?, start:
	Int, before: Int, count: Int) {
	onTextChanged(s.toString())
	}
	override fun afterTextChanged(s: Editable?) {}
	})
	}
	}

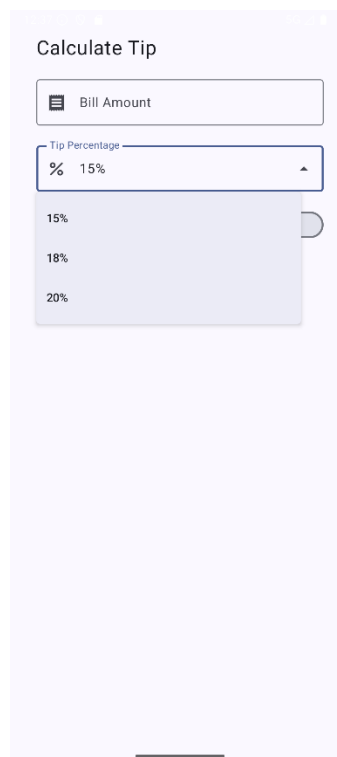
Tabel 3 Soruce Code Soal 1 - XML

B. Output Program



The screenshot shows a mobile application titled "Calculate Tip". It features a text input field labeled "Bill Amount" which is currently empty. Below it is a dropdown menu for "Tip Percentage" set to "15%". A toggle switch for "Round up tip?" is turned off. The result, "Tip Amount: \$0.00", is displayed in bold black text.

Gambar 4 Screenshot Hasil Jawaban Soal 1 - Compose



This screenshot shows the same "Calculate Tip" app, but with the "Tip Percentage" dropdown menu open. The menu lists three options: "15%", "18%", and "20%". The "Bill Amount" field remains empty, and the "Round up tip?" toggle is still off. The "Tip Amount: \$0.00" result is still visible at the bottom.

Gambar 5 Screenshot Hasil Jawaban Soal 1 - Compose

Calculate Tip

Bill Amount

2345

Tip Percentage

% 15%

Round up tip? ☐

Tip Amount: \$351.75

Gambar 6 Screenshot Hasil Jawaban Soal 1 - Compose

Calculate Tip

Bill Amount

2345

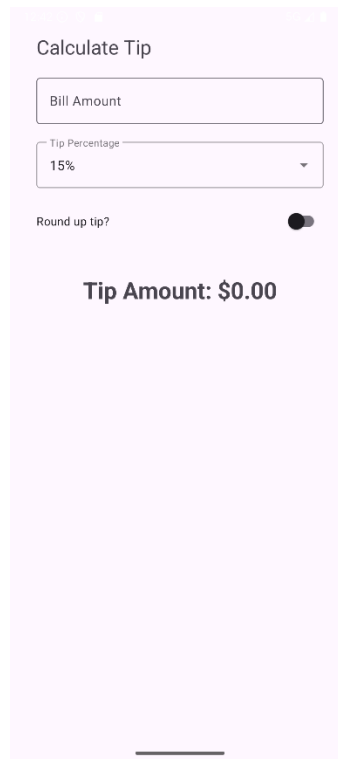
Tip Percentage

% 15%

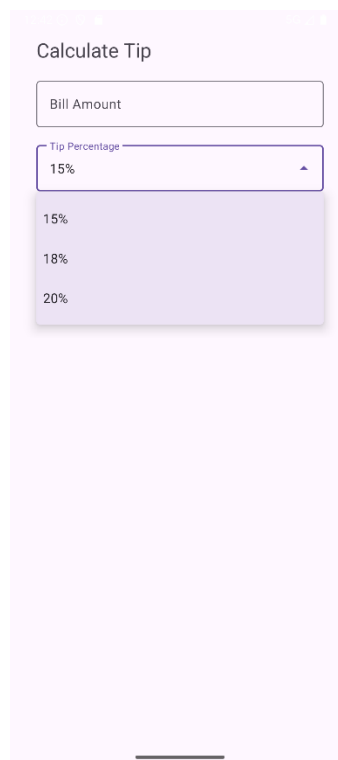
Round up tip? ☒

Tip Amount: \$352.00

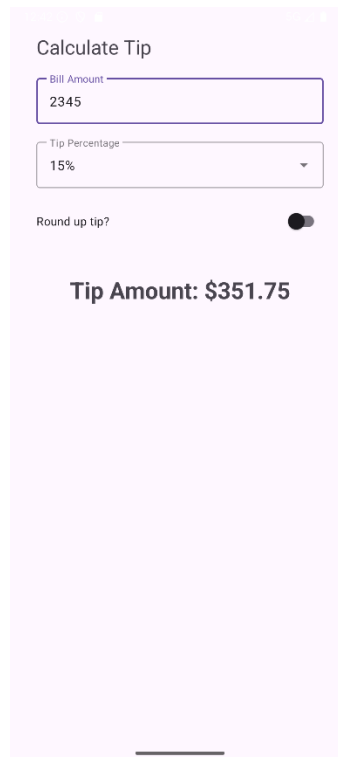
Gambar 7 Screenshot Hasil Jawaban Soal 1 – Compose



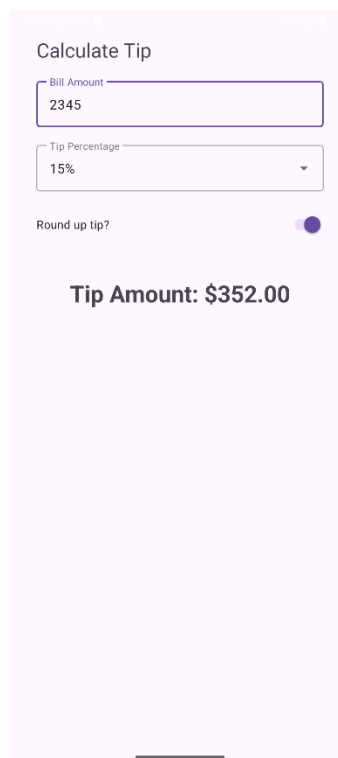
Gambar 8 Screenshot Hasil Jawaban Soal 1 - XML



Gambar 9 Screenshot Hasil Jawaban Soal 1 - XML



Gambar 10 Screenshot Hasil Jawaban Soal 1 - XML



Gambar 11 Screenshot Hasil Jawaban Soal 1 - XML

C. Pembahasan
MainActivity.kt / Compose :

- **Pada baris 21**, kelas MainActivity dideklarasikan sebagai turunan dari ComponentActivity, yang merupakan kelas dasar untuk activity yang menggunakan Jetpack Compose.
- **Pada baris 23**, di dalam fungsi onCreate, dipanggil setContent yang menjadi titik masuk untuk membangun UI dengan Jetpack Compose.
- **Pada baris 25**, KalkulatorTheme membungkus UI untuk menerapkan tema yang konsisten (warna, font, dll.) pada semua komponen di dalamnya.
- **Pada baris 26-29**, Surface digunakan sebagai container utama yang mengisi seluruh layar (fillMaxSize) dan menggunakan warna latar belakang dari tema.
- **Pada baris 30**, TipCalculatorScreen() dipanggil. Ini adalah fungsi Composable utama yang membangun seluruh tampilan dan logika dari kalkulator tip.
- **Pada baris 38**, @OptIn(ExperimentalMaterial3Api::class) digunakan karena beberapa komponen seperti ExposedDropDownMenuBox memerlukan persetujuan eksplisit untuk digunakan.
- **Pada baris 41**, var billAmountInput by remember { mutableStateOf("") } mendeklarasikan sebuah *state* untuk menyimpan input jumlah tagihan dari pengguna dalam bentuk String.
- **Pada baris 42**, var roundUp by remember { mutableStateOf(false) } mendeklarasikan *state* boolean untuk melacak apakah opsi "bulatkan tip" sedang aktif atau tidak, yang dikontrol oleh komponen Switch.
- **Pada baris 44**, val tipOptions dibuat sebagai sebuah List yang berisi opsi-opsi persentase tip yang tersedia.
- **Pada baris 45 & 46**, var expanded dan var selectedTipOption dideklarasikan sebagai *state*. expanded mengontrol apakah menu dropdown ditampilkan atau tidak, sementara selectedTipOption menyimpan opsi tip yang sedang dipilih pengguna.
- **Pada baris 48-50**, dilakukan proses kalkulasi. Nilai billAmountInput (String) diubah menjadi Double secara aman menggunakan toDoubleOrNull(). Persentase tip juga di-parsing dari String menjadi Double. Kemudian, fungsi calculateTip dipanggil dengan parameter-parameter ini untuk mendapatkan hasil akhir.
- **Pada baris 52**, Column digunakan untuk menyusun semua elemen UI secara vertikal. Atribut modifier digunakan untuk memberi padding, mengisi lebar layar, dan Arrangement.spacedBy memberi jarak vertikal yang seragam antar elemen.
- **Pada baris 59**, Text ditampilkan sebagai judul layar.
- **Pada baris 65**, EditNumberField(...) dipanggil. Ini adalah pemanggilan fungsi Composable yang dapat digunakan kembali untuk menampilkan input field jumlah tagihan.
- **Pada baris 71**, ExposedDropDownMenuBox digunakan sebagai container untuk membuat field teks yang bisa diklik untuk menampilkan menu dropdown.
- **Pada baris 76**, OutlinedTextField ditampilkan di dalam ExposedDropDownMenuBox. Field ini bersifat readOnly dan menampilkan selectedTipOption. Field ini juga memiliki leadingIcon (ikon persen) dan trailingIcon (ikon panah dropdown). Modifier .menuAnchor() penting untuk menandai TextField ini sebagai acuan posisi untuk ExposedDropDownMenu.

- **Pada baris 87**, `ExposedDropDownMenu` didefinisikan. Ini adalah menu yang akan muncul atau hilang berdasarkan *state* `expanded`.
- **Pada baris 91**, `tipOptions.forEach` digunakan untuk melakukan iterasi pada setiap opsi tip dan membuat sebuah `DropDownMenuItem` untuk masing-masing opsi. Saat sebuah item diklik, `selectedTipOption` diperbarui dan `expanded` di-set ke `false` untuk menutup menu.
- **Pada baris 102**, `RoundTheTipRow(...)` dipanggil. Ini adalah pemanggilan `Composable` lain yang dapat digunakan kembali untuk menampilkan baris "Round up tip?" beserta `Switch`-nya.
- **Pada baris 104**, `Spacer` digunakan untuk memberi ruang kosong vertikal.
- **Pada baris 107**, `Text` terakhir digunakan untuk menampilkan hasil kalkulasi tip (`Tip Amount: $tip`) dengan teks tebal dan ukuran font yang lebih besar.
- **Pada baris 116**, fungsi `Composable` ini dideklarasikan dengan parameter untuk label, nilai saat ini (`value`), fungsi yang akan dipanggil saat nilai berubah (`onValueChange`), dan ikon awalan (`leadingIcon`).
- **Pada baris 122**, `OutlinedTextField` digunakan sebagai elemen input. `keyboardOptions` diatur ke `KeyboardType.Number` untuk memastikan keyboard numerik yang muncul saat field ini disentuh.
- **Pada baris 133**, fungsi ini dideklarasikan dengan parameter untuk status `Switch` (`roundUp`) dan fungsi yang akan dipanggil saat statusnya berubah.
- **Pada baris 134**, `Row` digunakan untuk menyusun elemen secara horizontal.
- **Pada baris 138**, `Text` menampilkan label "Round up tip?".
- **Pada baris 139**, `Spacer` dengan modifier = `Modifier.weight(1f)` digunakan. Ini adalah trik untuk membuat `Spacer` mengisi semua ruang kosong yang tersedia, sehingga mendorong `Switch` ke ujung kanan baris.
- **Pada baris 140**, `Switch` ditampilkan. Status `checked`-nya terikat pada *state* `roundUp`, dan `onCheckedChange` akan memperbarui *state* tersebut saat `Switch` digeser.
- **Pada baris 144**, fungsi ini dideklarasikan sebagai `private` karena hanya digunakan di dalam file ini.
- **Pada baris 145**, perhitungan tip dasar dilakukan (`persentase / 100 * jumlah`).
- **Pada baris 146 & 147**, jika `roundUp` bernilai `true`, maka nilai tip dibulatkan ke atas ke bilangan bulat terdekat menggunakan `kotlin.math.ceil()`.
- **Pada baris 149**, `NumberFormat.getCurrencyInstance().format(tip)` digunakan untuk memformat hasil tip menjadi format mata uang lokal (misalnya, "Rp15.000" atau "\$15.00"), lalu mengembalikannya sebagai `String`.
- **Pada baris 152**, anotasi `@Preview` menandakan bahwa fungsi `Composable` ini akan dirender di dalam panel pratinjau `Android Studio`, memungkinkan developer melihat tampilan UI tanpa harus menjalankan aplikasi di perangkat atau emulator.

activity_main.xml / XML :

- **Pada baris 2**, LinearLayout digunakan sebagai layout utama yang menyusun semua elemen di dalamnya secara vertikal.
- **Pada baris 11**, TextView digunakan untuk menampilkan judul statis "Calculate Tip". **Pada baris 16**, com.google.android.material.textfield.TextInputLayout digunakan sebagai pembungkus untuk TextInputEditText. Komponen ini menyediakan fitur-fitur modern seperti label yang melayang (floating label) dan ruang untuk pesan error.
- **Pada baris 22**, com.google.android.material.textfield.TextInputEditText adalah field input teks sebenarnya tempat pengguna mengetikkan jumlah tagihan. inputType="numberDecimal" memastikan keyboard yang muncul adalah keyboard numerik.
- **Pada baris 28**, TextInputLayout kedua digunakan untuk menu dropdown. style diatur ke ...ExposedDropdownMenu untuk memberikan tampilan dropdown modern dari Material Design.
- **Pada baris 35**, AutoCompleteTextView ditempatkan di dalam TextInputLayout dropdown. Komponen ini, jika digabungkan dengan style di atas, akan berfungsi sebagai field teks yang menampilkan menu dropdown saat diklik. inputType="none" mencegah keyboard muncul saat field ini disentuh.
- **Pada baris 43**, com.google.android.material.switchmaterial.SwitchMaterial adalah komponen Switch dari Material Design yang digunakan untuk opsi "Round up tip?".
- **Pada baris 50**, TextView terakhir digunakan untuk menampilkan hasil akhir dari perhitungan tip. ID tip_result_text_view digunakan oleh kode Kotlin untuk memperbarui teks di dalamnya. textStyle="bold" dan ukuran font yang besar digunakan untuk memberikan penekanan visual pada hasil.

MainActivity.kt / XML :

- **Pada baris 11**, kelas MainActivity dideklarasikan sebagai turunan dari AppCompatActivity, kelas dasar untuk Activity pada model View-XML.
- **Pada baris 13**, dideklarasikan variabel binding yang akan digunakan untuk mengakses semua komponen (View) dari file activity_main.xml secara aman (View Binding).
- **Pada baris 15**, di dalam fungsi onCreate yang pertama kali dijalankan, binding diinisialisasi dengan "mengubah" layout XML menjadi objek Kotlin.
- **Pada baris 17**, setContentView(binding.root) menetapkan tampilan dari activity ini menggunakan layout yang sudah di-binding.
- **Pada baris 19**, tipOptions diinisialisasi dengan mengambil array string dari file strings.xml. Array ini berisi opsi-opsi persentase tip.
- **Pada baris 20**, sebuah ArrayAdapter dibuat. Adapter ini berfungsi sebagai jembatan yang menghubungkan data (tipOptions) dengan komponen UI yang bisa menampilkannya (dalam kasus ini, AutoCompleteTextView).
- **Pada baris 21**, setAdapter(adapter) digunakan untuk memasang adapter tersebut ke tipPercentageAutocomplete, sehingga komponen tersebut tahu data apa yang harus ditampilkan sebagai opsi dropdown.
- **Pada baris 22**, setText(tipOptions[0], false) digunakan untuk mengatur nilai default yang tampil di field dropdown. Argumen false mencegah filter dropdown berjalan saat nilai diatur secara programatik.

- **Pada baris 24 & 25**, `setupListeners()` dan `calculateAndDisplayTip()` dipanggil untuk pertama kalinya untuk memasang semua "pendengar event" dan menampilkan nilai tip awal (yaitu \$0.00).
- **Pada baris 29**, sebuah listener ditambahkan ke `billAmountEditText` menggunakan fungsi ekstensi kustom. Setiap kali teks di dalam field ini berubah, fungsi `calculateAndDisplayTip()` akan otomatis dipanggil.
- **Pada baris 30**, `setOnCheckedChangeListener` dipasang pada `roundUpSwitch`. Setiap kali status switch berubah (dicentang atau tidak), `calculateAndDisplayTip()` akan dipanggil.
- **Pada baris 32**, `setOnItemClickListener` dipasang pada `tipPercentageAutocomplete`. Listener ini akan aktif saat pengguna memilih salah satu item dari menu dropdown, yang kemudian akan memanggil `calculateAndDisplayTip()`.
- **Pada baris 38 & 39**, nilai-nilai terbaru dari `billAmountEditText` dan `tipPercentageAutocomplete` dibaca dan diubah menjadi `String`.
- **Pada baris 41-43**, nilai-nilai `String` tersebut di-parsing menjadi tipe data yang sesuai (`Double` dan `Boolean`) secara aman.
- **Pada baris 45-48**, logika perhitungan tip dilakukan. Ini sama persis dengan versi `Compose`, termasuk opsi untuk membulatkan ke atas.
- **Pada baris 50**, hasil tip diformat menjadi format mata uang lokal (misal: "Rp15.000").
- **Pada baris 51**, properti `text` dari `tipResultTextView` diperbarui untuk menampilkan hasil tip yang sudah diformat. Inilah langkah "menulis" kembali ke UI.
- **Pada baris 54**, sebuah fungsi ekstensi dideklarasikan untuk kelas `android.widget.EditText`. Ini artinya, semua objek `EditText` bisa memanggil fungsi ini seolah-olah fungsi ini adalah bawaannya.
- **Pada baris 55-61**, implementasi `TextWatcher` yang biasanya panjang dan butuh tiga fungsi *override* (`before`, `on`, `after`) disederhanakan. Fungsi ini hanya peduli pada `onTextChanged` dan mengeksposnya sebagai sebuah lambda (`onTextChanged: (String) -> Unit`) yang lebih bersih dan mudah digunakan.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/AdiYaus/Praktikum-PemrogramanMobile.git>

SOAL 2

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

A. Jawaban

Perbedaan, Kelebihan, dan Kekurangan: XML vs Jetpack Compose

Ini adalah bagian terpenting dari soal praktikummu. Mari kita bandingkan keduanya.

Paradigma Pemrograman

- **XML: Imperatif** (memerintah). Kamu mendefinisikan layout di XML, lalu di kode Kotlin, kamu *memerintah* setiap komponen untuk berubah. Contoh: `binding.tipResultTextView.text = "..."`. Kamu secara manual mencari komponen dan mengubah propertinya.
- **Jetpack Compose: Deklaratif**. Kamu *mendeklarasikan* atau mendeskripsikan bagaimana UI seharusnya terlihat untuk *state* (data) tertentu. Kamu tidak pernah mengubah UI secara langsung. Sebaliknya, kamu mengubah *state*-nya (`var amount by remember { ... }`), dan Compose akan secara otomatis memperbarui UI untukmu.

Kelebihan dan Kekurangan

Jetpack Compose

- **Kelebihan:**
 - **Kode Lebih Sedikit:** Secara signifikan mengurangi jumlah kode yang perlu ditulis.
 - **Intuitif:** Pendekatan deklaratif membuat alur data lebih mudah dipahami. "UI adalah cerminan dari data."
 - **Pengembangan Cepat:** Perubahan kecil pada UI tidak memerlukan build ulang yang kompleks. Fitur *preview* sangat cepat dan interaktif.
 - **Powerful:** Sangat mudah membuat animasi dan UI kustom yang kompleks.
- **Kekurangan:**
 - **Kurva Belajar:** Membutuhkan perubahan pola pikir bagi developer yang sudah sangat terbiasa dengan XML.
 - **Ekosistem Baru:** Meskipun berkembang pesat, beberapa library pihak ketiga mungkin belum sepenuhnya terintegrasi atau tersedia untuk Compose.

XML + Kotlin

- **Kelebihan:**

- **Mapan (Mature):** Telah ada selama bertahun-tahun, sangat stabil, dan didukung oleh banyak sekali library dan contoh di komunitas.
- **Pemisahan yang Jelas:** Beberapa developer menyukai pemisahan ketat antara file desain (XML) dan file logika (Kotlin).
- **Visual Editor:** Editor layout XML di Android Studio sangat powerfull dan mudah digunakan untuk mendesain UI secara *drag-and-drop*.

- **Kekurangan:**

- **Kode Boilerplate:** Membutuhkan banyak kode "perekat" seperti ViewBinding dan listener yang membuat kode lebih panjang.
- **Manajemen State Manual:** Mengelola state di UI yang kompleks bisa menjadi rumit dan rentan terhadap kesalahan (misalnya, lupa memperbarui salah satu TextView).
- **Verbosity:** XML bisa menjadi sangat panjang dan sulit dibaca untuk layout yang rumit.

Singkatnya, **Jetpack Compose adalah masa depan pengembangan UI Android** karena lebih modern, efisien, dan ringkas. Namun, **XML masih sangat relevan** dan akan tetap ada untuk waktu yang lama, terutama dalam proyek-proyek besar yang sudah ada.

