LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2

ANDROID LAYOUT WITH COMPOSE

Oleh:

Adrian Bintang Saputera NIM. 2310817110006

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 2025

LEMBAR PENGESAHAN

LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 2

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

Nama Praktikan : Adrian Bintang Saputera
NIM : 2310817110006

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 199307032019030101 1

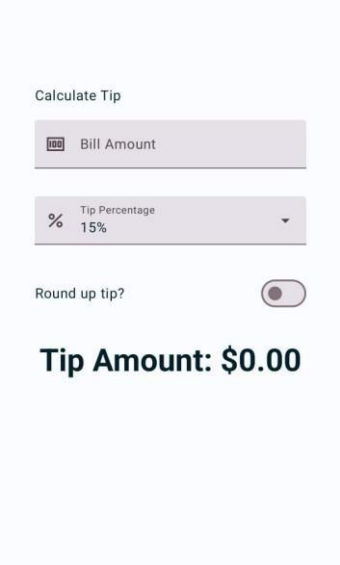
DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

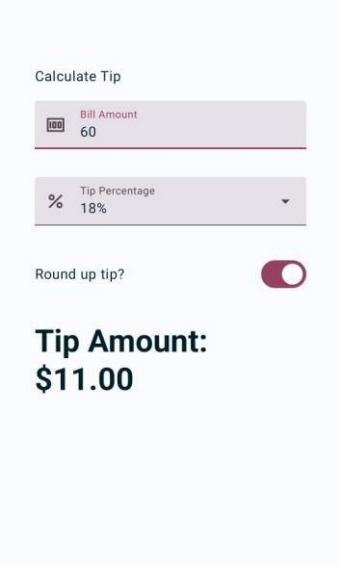
SOAL

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.



Gambar 1. Tampilan Awal Aplikasi

Gambar 2. Tampilan Pilihan Persentase Tip



Gambar 3. Tampilan Aplikasi Setelah Dijalankan

2. Perbedaan utama antara implementasi XML dan Jetpack Compose terletak pada cara penyusunan dan pengelolaan antarmuka pengguna (UI). XML menggunakan pendekatan deklaratif, di mana tampilan UI didefinisikan dalam file terpisah dengan elemen-elemen UI yang statis, sementara Jetpack Compose memungkinkan pembuatan UI secara langsung dalam kode Kotlin dengan pendekatan deklaratif dan reaktif. XML sudah lama digunakan dalam pengembangan Android dan memiliki banyak dukungan alat dan dokumentasi, namun sering kali lebih verbose dan kurang fleksibel dalam

pembaruan UI. Di sisi lain, Jetpack Compose lebih ringkas, mudah dibaca, dan fleksibel, dengan kemampuan untuk merespons perubahan data secara otomatis tanpa perlu pengaturan tambahan. Namun, Jetpack Compose masih relatif baru, dan meskipun terus berkembang, ada kurva belajar bagi pengembang yang terbiasa dengan XML. Meskipun XML lebih stabil dan cocok untuk aplikasi besar atau yang sudah mapan, Jetpack Compose lebih cocok untuk aplikasi baru yang ingin memanfaatkan fungsionalitas modern dan meningkatkan produktivitas pengembang.

A. Source Code

XML:

MainActivity.kt

Table 1. Source Code Jawaban Soal 1 XML

```
1  package com.example.tipsxml
2
3  import android.os.Bundle
4  import android.text.Editable
5  import android.text.TextWatcher
6  import android.view.View
7  import androidx.appcompat.app.AppCompatActivity
   import
8  com.example.tipsxml.databinding.ActivityMainBinding
9  import java.text.NumberFormat
10 import java.util.Locale
11 import kotlin.math.ceil
12
13 class MainActivity : AppCompatActivity() {
14
15     private lateinit var binding: ActivityMainBinding
16     private var amount: String = ""
17     private var tipOption: String = "15%"
18     private var roundUp: Boolean = false
19
20     override fun onCreate(savedInstanceState: Bundle?)
   {
21         super.onCreate(savedInstanceState)
22         binding =
23 ActivityMainBinding.inflate(layoutInflater)
24         setContentView(binding.root)
25
```

```
        val tipOptions = listOf("15%", "18%", "20%")
        val adapter = android.widget.ArrayAdapter(this,
android.R.layout.simple_dropdown_item_1line,
tipOptions)

binding.autoCompleteTextView.setAdapter(adapter)
        binding.autoCompleteTextView.setText(tipOption,
false)


binding.textInputEditText.addTextChangedListener(object
: TextWatcher {
            override fun afterTextChanged(s: Editable?)
{
                amount = s.toString()
                updateTip()
            }
            override fun beforeTextChanged(s:
CharSequence?, start: Int, count: Int, after: Int) {}
            override fun onTextChanged(s:
CharSequence?, start: Int, before: Int, count: Int) {}
        })


binding.autoCompleteTextView.setOnItemClickListener {
parent, _, position, _ ->
            tipOption =
parent.getItemAtPosition(position) as String
            updateTip()
        }

        binding.switch1.setOnCheckedChangeListener { _,
isChecked ->
            roundUp = isChecked
            updateTip()
        }

        updateTip()
    }

    private fun updateTip() {
        val amountDouble = amount.toDoubleOrNull() ?:
0.0
        val tipPercent = when (tipOption) {
            "15%" -> 0.15
```

```
66              "18%" -> 0.18
67              "20%" -> 0.20
                else -> 0.15
68          }
69
70          var tip = amountDouble * tipPercent
71          if (roundUp) tip = ceil(tip)
72
73          val formattedTip =
74  NumberFormat.getCurrencyInstance(Locale.US).format(tip)

            if (formattedTip == "$0.00") {
                binding.tipAmount.text = "Tip Amount:
    $0.00"
                binding.tipAmount.textAlignment =
    View.TEXT_ALIGNMENT_CENTER
            } else {
                binding.tipAmount.text = "Tip
    Amount:\n$formattedTip"
                binding.tipAmount.textAlignment =
    View.TEXT_ALIGNMENT_VIEW_START
            }
        }
    }
```

activity_main.xml

Table 2. Source Code Jawaban Soal 1 XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/and
4   roid"
5       xmlns:app="http://schemas.android.com/apk/res-
6   auto"
7       xmlns:tools="http://schemas.android.com/tools"
8       android:id="@+id/main"
9       android:layout_width="match_parent"
1       android:layout_height="match_parent"
0       tools:context=".MainActivity">
1
1       <TextView
1           android:id="@+id/textView"
2           android:layout_width="wrap_content"
1           android:layout_height="wrap_content"
3           android:layout_marginStart="48dp"
```

```
        android:layout_marginTop="148dp"
        android:text="@string/calculate_tip"
        android:textSize="16sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


<com.google.android.material.textfield.TextInputLayout
        android:id="@+id/billAmount"
style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="46dp"
        android:layout_marginTop="12dp"
        android:layout_marginEnd="46dp"
        android:hint="@string/bill_amount"
        app:startIconDrawable="@drawable/money"
app:layout_constraintTop_toBottomOf="@id/textView"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent">


<com.google.android.material.textfield.TextInputEditText
            android:id="@+id/textInputEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="numberDecimal" />

</com.google.android.material.textfield.TextInputLayout>


<com.google.android.material.textfield.TextInputLayout
        android:id="@+id/tipPercentage"
style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox.ExposedDropdownMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_marginStart="46dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="46dp"
        android:hint="@string/how_was_the_service"
        app:startIconDrawable="@drawable/percent"

app:layout_constraintTop_toBottomOf="@id/billAmount"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <AutoCompleteTextView
            android:id="@+id/autoCompleteTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="none"
            tools:ignore="LabelFor" />


</com.google.android.material.textfield.TextInputLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/round_up_tip"

app:layout_constraintTop_toBottomOf="@id/tipPercentage"
        app:layout_constraintStart_toStartOf="parent"
        android:layout_marginStart="46dp"
        android:layout_marginTop="28dp"
        android:textSize="15sp"
        tools:ignore="NotSibling" />

    <TextView
        android:id="@+id/tipAmount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="84dp"
        android:text="@string/tip_amount"
        android:textAlignment="viewStart"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_marginStart="46dp"
        app:layout_constraintStart_toStartOf="parent"
```

```
                                                    
app:layout_constraintTop_toBottomOf="@+id/tipPercenta
ge" />
                                                    
    <Switch
        android:id="@+id/switch1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:ignore="UseSwitchCompatOrMaterialXml"
        android:layout_marginTop="30dp"
                                                    
app:layout_constraintEnd_toEndOf="@id/tipPercentage"
                                                    
app:layout_constraintTop_toBottomOf="@id/tipPercentag
e">
    </Switch>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Jetpack Compose:

MainActivity.kt

Table 3. Source Code Jawaban Soal 1 Jetpack Compose

```
1  package com.example.tipscompose
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.annotation.DrawableRes
8  import androidx.annotation.StringRes
9  import androidx.compose.foundation.layout.*
10 import
11 androidx.compose.foundation.rememberScrollState
12 import
13 androidx.compose.foundation.text.KeyboardOptions
14 import androidx.compose.foundation.verticalScroll
15 import androidx.compose.material3.*
16 import androidx.compose.runtime.*
17 import androidx.compose.ui.Alignment
18 import androidx.compose.ui.Modifier
19 import androidx.compose.ui.res.painterResource
20 import androidx.compose.ui.res.stringResource
21 import androidx.compose.ui.text.input.ImeAction
22 import androidx.compose.ui.text.input.KeyboardType
```

```
23 import androidx.compose.ui.tooling.preview.Preview
24 import androidx.compose.ui.unit.dp
25 import
26 com.example.tipscompose.ui.theme.TipsComposeTheme
27 import java.text.NumberFormat
28
29 class MainActivity : ComponentActivity() {
30     override fun onCreate(savedInstanceState:
31 Bundle?) {
32         enableEdgeToEdge()
33         super.onCreate(savedInstanceState)
34         setContent {
35             TipsComposeTheme {
36                 Surface(
37                     modifier =
38 Modifier.fillMaxSize(),
39                 ) {
40                     TipTimeLayout()
41                 }
42             }
43         }
44     }
45 }
46
47 @Composable
48 fun TipTimeLayout() {
49     var amountInput by remember { mutableStateOf("")
50 }
51     var tipInput by remember { mutableStateOf("15")
52 }
53     var roundUp by remember { mutableStateOf(false)
54 }
55
56     val amount = amountInput.toDoubleOrNull() ?: 0.0
57     val tipPercent = tipInput.toDoubleOrNull() ?: 0.0
58     val tip = calculateTip(amount, tipPercent,
59 roundUp)
60
61     Column(
62         modifier = Modifier
63             .statusBarsPadding()
64             .padding(horizontal = 40.dp)
65             .verticalScroll(rememberScrollState())
66             .safeDrawingPadding(),
67         horizontalAlignment =
```

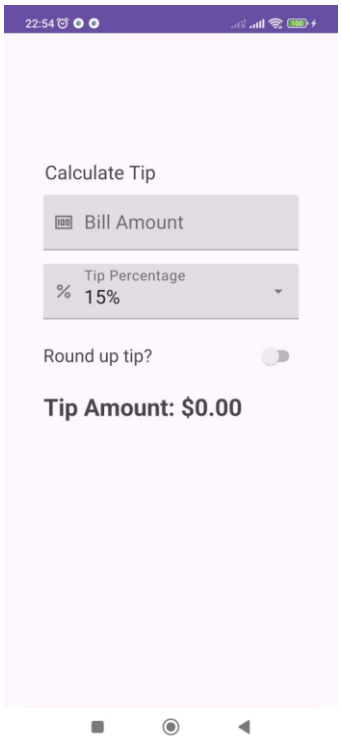
```
68  Alignment.CenterHorizontally,
69          verticalArrangement   =   Arrangement.Center
70      )                                                {
71          Text(
72              text                                     =
73  stringResource(R.string.calculate_tip),
74              modifier            =            Modifier
75                  .padding(bottom = 16.dp, top = 40.dp)
76                  .align(alignment  =  Alignment.Start)
77          )
78          EditNumberField(
79              label       =       R.string.bill_amount,
80              leadingIcon     =     R.drawable.money,
81              keyboardOptions                          =
82  KeyboardOptions.Default.copy(
83                  keyboardType  =  KeyboardType.Number,
84                  imeAction      =      ImeAction.Next
85              ),
86              value           =           amountInput,
87              onValueChanged  =  {  amountInput  =  it  },
88              modifier   =   Modifier.padding(bottom   =
89  32.dp).fillMaxWidth(),
90          )
91          TipPercentageDropdown(
92              selectedTip          =          tipInput,
93              onTipSelected  =  {  tipInput  =  it  },
94              modifier   =   Modifier.padding(bottom   =
95  32.dp).fillMaxWidth()
96          )
97          RoundTheTipRow(
98              roundUp             =             roundUp,
99              onRoundUpChanged  =  {  roundUp  =  it  },
10              modifier   =   Modifier.padding(bottom   =
0   32.dp)
10          )
1           Text(
10              text                                     =
2   stringResource(R.string.tip_amount,            tip),
10              style                                    =
3   MaterialTheme.typography.displaySmall
10          )
4           Spacer(modifier   =   Modifier.height(150.dp))
10      }
5   }
10
```

```
@Composable
fun EditNumberField(
    @StringRes label: Int,
    @DrawableRes leadingIcon: Int,
    keyboardOptions: KeyboardOptions,
    value: String,
    onValueChanged: (String) -> Unit,
    modifier: Modifier = Modifier
) {
    TextField(
        value = value,
        singleLine = true,
        leadingIcon = { Icon(painter = painterResource(id = leadingIcon), null) },
        modifier = modifier,
        onValueChange = onValueChanged,
        label = { Text(stringResource(label)) },
        keyboardOptions = keyboardOptions
    )
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TipPercentageDropdown(
    selectedTip: String,
    onTipSelected: (String) -> Unit,
    modifier: Modifier = Modifier
) {
    val options = listOf("15", "18", "20")
    var expanded by remember { mutableStateOf(false) }

    ExposedDropdownMenuBox(
        expanded = expanded,
        onExpandedChange = { expanded = !expanded },
        modifier = modifier
    ) {
        TextField(
            value = "$selectedTip%",
            onValueChange = {},
            readOnly = true,
            label = {
Text(stringResource(R.string.how_was_the_service))
},
            leadingIcon = { Icon(painter =
```

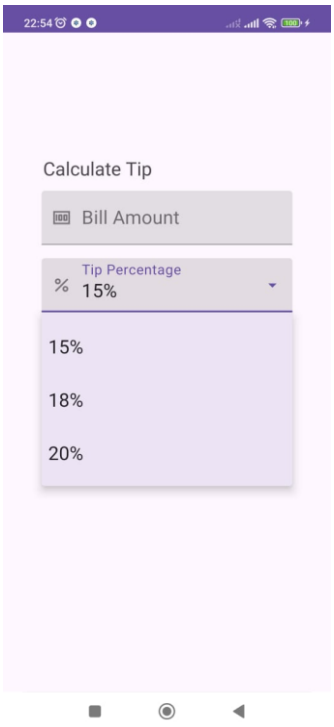
```
painterResource(id = R.drawable.percent),
contentDescription = null) },
            trailingIcon = {
ExposedDropdownMenuDefaults.TrailingIcon(expanded =
expanded) },
            modifier = Modifier
.menuAnchor(MenuAnchorType.PrimaryEditable, enabled
= true)
                .fillMaxWidth()
        )
        ExposedDropdownMenu(
            expanded = expanded,
            onDismissRequest = { expanded = false }
        ) {
            options.forEach { option ->
                DropdownMenuItem(
                    text = { Text("$option%") },
                    onClick = {
                        onTipSelected(option)
                        expanded = false
                    }
                )
            }
        }
    }
}

@Composable
fun RoundTheTipRow(
    roundUp: Boolean,
    onRoundUpChanged: (Boolean) -> Unit,
    modifier: Modifier = Modifier
) {
    Row(
        modifier = modifier.fillMaxWidth(),
        verticalAlignment =
Alignment.CenterVertically
    ) {
        Text(text =
stringResource(R.string.round_up_tip))
        Switch(
            modifier = Modifier
                .fillMaxWidth()
                .wrapContentWidth(Alignment.End),
```

```
            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(java.util.Locale.U
S).format(tip)
}

@Preview(showBackground = true)
@Composable
fun TipTimeLayoutPreview() {
    TipsComposeTheme {
        TipTimeLayout()
    }
}
```

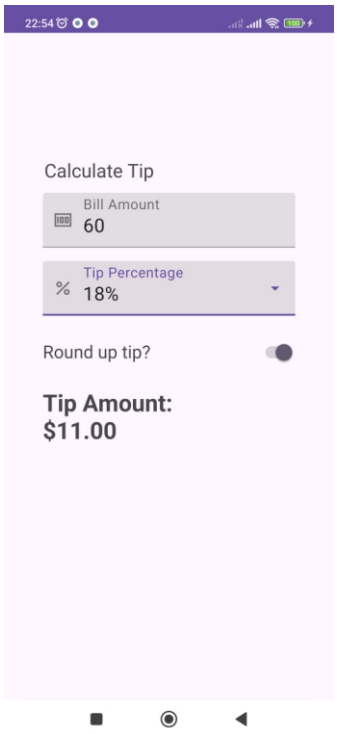
B. Output Program



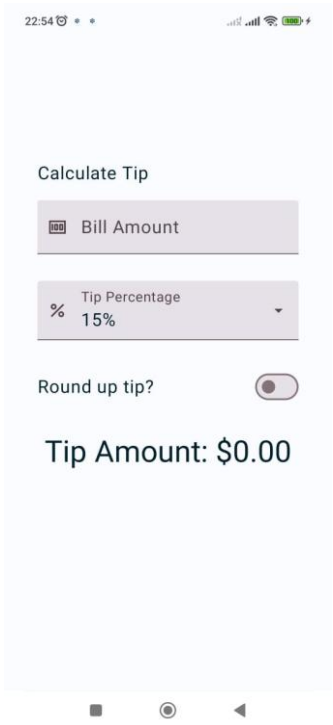
Gambar 4. Tampilan Awal Aplikasi XML



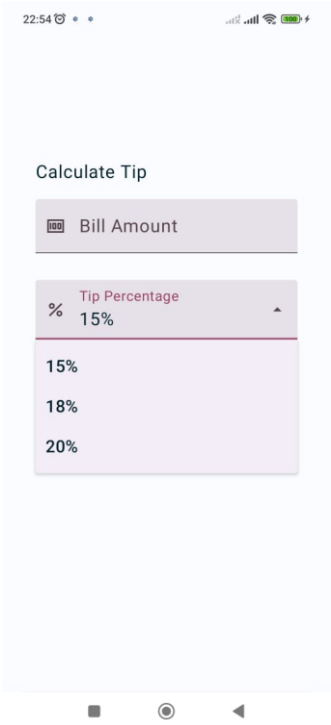
Gambar 5. Tampilan Pilihan Presentase Tip XML



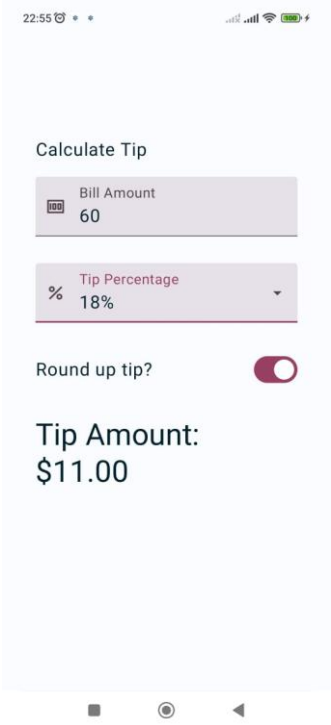
Gambar 6. Tampilan Aplikasi Setelah Dijalankan Aplikasi XML



Gambar 7. Tampilan Awal Aplikasi Jetpack Compose



Gambar 8. Tampilan Aplikasi Setelah Dijalankan Aplikasi Jetpack Compose



Gambar 9. Tampilan Aplikasi Setelah Dijalankan Aplikasi Jetpack Compose

C. Pembahasan

XML:

MainActivity.kt:

Kode ini diawali dengan mendeklarasikan package com.example.tipsxml dan mengimpor berbagai library Android yang dibutuhkan untuk membangun aplikasi tip calculator ini, termasuk pustaka untuk mengatur tampilan, teks, dan format angka.

Kemudian, kelas MainActivity didefinisikan sebagai turunan dari AppCompatActivity, yang menjadi titik masuk utama saat aplikasi dibuka. Di dalam kelas ini, beberapa variabel dideklarasikan: binding untuk menghubungkan layout XML menggunakan ViewBinding, amount untuk menyimpan input jumlah uang dari pengguna, tipOption untuk menyimpan pilihan persentase tip, dan roundUp untuk menentukan apakah tip akan dibulatkan ke atas atau tidak.

Pada fungsi onCreate, binding diinisialisasi dan layout ditetapkan menggunakan setContentView. Daftar opsi tip seperti 15%, 18%, dan 20% disiapkan dan dimasukkan ke dalam sebuah ArrayAdapter, yang kemudian digunakan untuk mengisi komponen AutoCompleteTextView. Nilai awal pilihan tip juga diatur ke 15%.

Sebuah TextWatcher dipasang pada kolom input jumlah uang. Setelah teks berubah, nilai amount diperbarui dan fungsi updateTip() dipanggil untuk menghitung ulang tip. Dua fungsi lainnya dalam TextWatcher dibiarkan kosong karena tidak diperlukan.

Ketika pengguna memilih salah satu opsi tip dari dropdown, nilai tipOption diperbarui dan perhitungan tip diperbarui juga. Switch untuk pembulatan tip juga dipasang dengan listener yang akan mengubah nilai roundUp dan memicu ulang perhitungan.

Fungsi updateTip() menangani logika utama perhitungan tip. Input dari pengguna dikonversi menjadi angka desimal, dan nilai persen ditentukan berdasarkan pilihan pengguna. Tip dihitung dengan mengalikan jumlah dengan persentase. Jika pembulatan diaktifkan, maka nilai tip dibulatkan ke atas. Hasilnya kemudian diformat ke dalam format dolar AS menggunakan NumberFormat.

Terakhir, hasil tip ditampilkan ke pengguna. Jika hasilnya nol, teks ditampilkan dengan penyesuaian posisi ke tengah. Jika ada nilai, teks ditampilkan dengan pemisahan baris dan rata kiri.

Activity_main.xml:

Layout ini menggunakan ConstraintLayout sebagai wadah utama yang memungkinkan setiap elemen UI ditempatkan relatif terhadap elemen lainnya atau terhadap parent-nya dengan fleksibilitas tinggi.

Bagian pertama adalah TextView yang menampilkan judul atau petunjuk seperti "Calculate Tip". Letaknya diatur agar berada di bagian atas layar dengan margin kiri dan atas yang cukup besar, dan teksnya menggunakan ukuran sedang.

Setelah itu, ada TextInputLayout pertama yang berisi TextInputEditText. Ini adalah kolom tempat pengguna memasukkan jumlah tagihan (bill amount). Komponen ini dilengkapi ikon bergambar uang dan memiliki style FilledBox agar terlihat modern dan material-friendly.

Di bawahnya terdapat TextInputLayout kedua, yang menggunakan style dropdown. Komponen ini menampilkan pilihan persentase tip (misalnya 15%, 18%, 20%) menggunakan AutoCompleteTextView. Di sini juga terdapat ikon dengan gambar persen yang menunjukkan konteks input.

Berikutnya adalah TextView lagi yang berfungsi memberi label untuk switch pembulatan tip, seperti "Round up tip". Letaknya tepat di bawah dropdown persentase tip, dengan margin atas yang sedang dan rata kiri.

Kemudian ada TextView untuk menampilkan hasil tip yang dihitung. Teks ini diberi ukuran yang lebih besar dan dicetak tebal untuk menonjolkan hasilnya. Letaknya juga berada di bawah komponen pilihan persentase tip.

Terakhir adalah Switch, yaitu tombol geser yang digunakan pengguna untuk mengaktifkan atau menonaktifkan pembulatan nilai tip ke atas. Letaknya sejajar dengan label "Round up tip" dan mengikuti margin kanan dari elemen sebelumnya.

Secara keseluruhan, layout ini dirancang agar sederhana dan intuitif, memudahkan pengguna dalam memasukkan jumlah, memilih tip, dan melihat hasilnya langsung.

Jetpack Compose:

MainActivity.kt:

Berikut penjelasan per bagian dari kode Compose di atas tanpa menyebutkan ulang potongan kodenya:

Bagian awal adalah deklarasi MainActivity, yang merupakan ComponentActivity. Di dalam onCreate, layout aplikasi diatur menggunakan fungsi setContent, dan tema TipsComposeTheme dibungkus dalam sebuah Surface yang mengisi seluruh ukuran layar. Fungsi utama yang ditampilkan adalah TipTimeLayout.

Dalam fungsi TipTimeLayout, terdapat tiga state yang dideklarasikan untuk menyimpan input pengguna: jumlah tagihan (amountInput), persentase tip (tipInput), dan apakah tip perlu dibulatkan (roundUp). Dari input ini, dihitung nilai tip akhir melalui fungsi calculateTip.

Layout disusun secara vertikal menggunakan Column, dan diatur agar bisa di-scroll jika kontennya melebihi layar. Padding digunakan agar elemen tidak menempel di sisi layar. Komponen-komponen yang ditampilkan meliputi:

- Teks judul “Calculate Tip”
- Kolom input untuk jumlah tagihan dengan ikon uang di sebelah kiri, menggunakan TextField dan EditNumberField
- Dropdown menu untuk memilih persentase tip menggunakan ExposedDropdownMenuBox, yang memungkinkan pengguna memilih nilai dari daftar opsi (15, 18, 20)
- Baris untuk switch pembulatan, terdiri dari label dan toggle Switch
- Hasil perhitungan tip yang ditampilkan dalam teks besar
- Spacer di bagian bawah agar ada ruang tambahan setelah elemen-elemen utama

EditNumberField adalah komponen kustom untuk membuat kolom input angka, menerima parameter seperti label, ikon, opsi keyboard, nilai saat ini, dan callback perubahan.

TipPercentageDropdown menampilkan dropdown menu yang dibuka atau ditutup saat pengguna mengetuk kolom. Nilai terpilih ditampilkan sebagai teks. Saat pengguna memilih salah satu opsi, state tipInput diperbarui.

RoundTheTipRow adalah baris horizontal dengan label teks dan tombol Switch untuk mengaktifkan pembulatan ke atas.

Fungsi calculateTip menerima nilai jumlah dan persentase tip. Tip dihitung dari perkalian, dan jika roundUp aktif, nilainya dibulatkan ke atas. Hasil akhirnya diformat menjadi mata uang dalam format dolar AS.

Terakhir, fungsi TipTimeLayoutPreview adalah anotasi Compose Preview yang memungkinkan tampilan layout divisualisasikan di Android Studio tanpa perlu menjalankan aplikasi.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/MadeByBintang/PraktikumMobile