LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2

ANDROID LAYOUT WITH COMPOSE

Oleh:

Andra Braputra Akbar Saleh NIM. 2310817210001

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 : Andra Braputra Akbar Saleh
NIM : 2310817210001

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Salsabila Syifa NIM. 2010817320004	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

XML

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.

A. Source Code

MainActivity.kt

```
1  package com.example.tipxml
2
3  import android.os.Bundle
4  import android.text.Editable
5  import android.text.TextWatcher
6  import android.view.View
7  import android.widget.AdapterView
8  import android.widget.ArrayAdapter
9  import androidx.appcompat.app.AppCompatActivity
10 import com.example.tipxml.databinding.ActivityMainBinding
11 import java.text.NumberFormat
12 import kotlin.math.ceil
13
14 class MainActivity : AppCompatActivity() {
15
16     private lateinit var binding: ActivityMainBinding
17     private val tipOptions = listOf("10%", "15%", "20%")
18
19     override fun onCreate(savedInstanceState: Bundle?) {
20         super.onCreate(savedInstanceState)
21         binding = ActivityMainBinding.inflate(layoutInflater)
22         setContentView(binding.root)
23         val           adapter           =           ArrayAdapter(this,
24 android.R.layout.simple_spinner_dropdown_item, tipOptions)
25
26         binding.spinner.adapter = adapter
27
```

```
28          // Spinner item selected inline listener
29          binding.spinner.onItemSelectedListener     =     object     :
30  AdapterView.OnItemSelectedListener {
31              override fun onItemSelected(
32                  parent: AdapterView<*>,
33                  view: View?,
34                  position: Int,
35                  id: Long
36              ) {
37                  updateTip()
38              }
39
40              override fun onNothingSelected(parent: AdapterView<*>) {}
41          }
42
43          // TextWatcher for input changes
44          binding.etBill.addTextChangedListener(object : TextWatcher {
45              override fun afterTextChanged(s: Editable?) = updateTip()
46              override  fun  beforeTextChanged(s:  CharSequence?,  start:
47  Int, count: Int, after: Int) {}
48              override fun onTextChanged(s: CharSequence?, start: Int,
49  before: Int, count: Int) {}
50          })
51
52          // Switch listener
53          binding.switchRound.setOnCheckedChangeListener  {  _,  _  ->
54  updateTip() }
55      }
56
57      private fun updateTip() {
58          val  amount  =  binding.etBill.text.toString().toDoubleOrNull()
59  ?: 0.0
60          val tipText = binding.spinner.selectedItem.toString()
61          val  tipPercent  =  tipText.removeSuffix("%").toDoubleOrNull()
62  ?: 15.0
63          val roundUp = binding.switchRound.isChecked
64          val tip = calculateTip(amount, tipPercent, roundUp)
65
            binding.tvTotalTip.text = getString(R.string.tip_amount, tip)
        }

        private  fun  calculateTip(amount:  Double,  tipPercent:  Double,
    roundUp: Boolean): String {
            var tip = tipPercent / 100 * amount
            if (roundUp) {
                tip = ceil(tip)
            }
            return NumberFormat.getCurrencyInstance().format(tip)
        }
    }
```

Tabel 1. Source Code XML

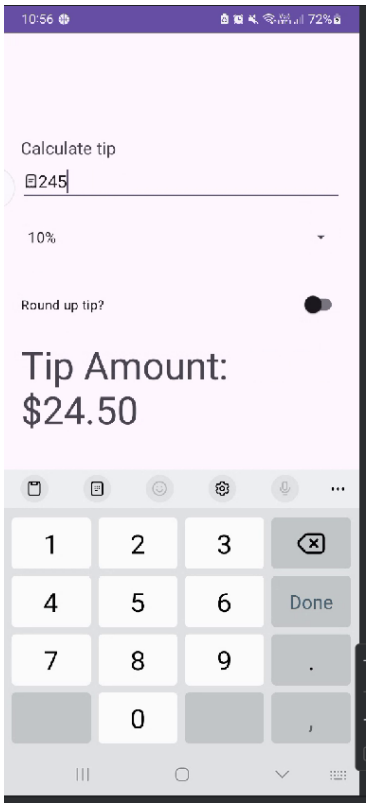
activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textViewLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="120dp"
        android:text="@string/calculate_tip"
        android:textSize="18sp"
        app:layout_constraintBottom_toTopOf="@id/etBill"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.067"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

    <EditText
        android:id="@+id/etBill"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_marginTop="2dp"
        android:autofillHints=""
        android:hint="@string/bill_amount"
        android:drawableStart="@drawable/receipt_1_"
        android:inputType="numberDecimal"

app:layout_constraintTop_toBottomOf="@+id/textViewLabel"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintWidth_percent="0.9"/>

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_marginTop="16dp"
        app:layout_constraintTop_toBottomOf="@id/etBill"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintWidth_percent="0.9"
        />
```

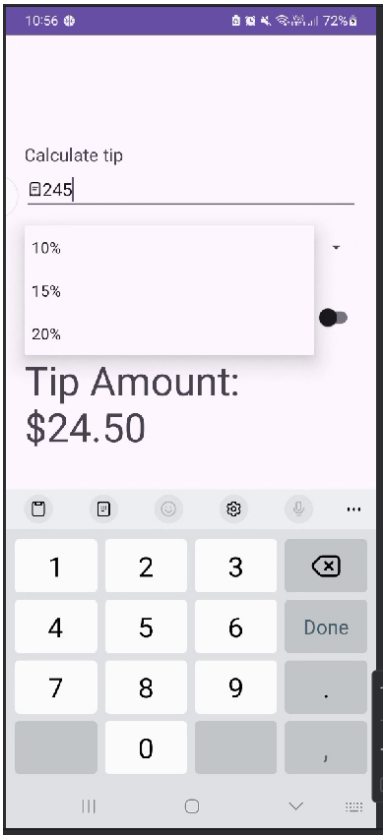
```
    <androidx.appcompat.widget.SwitchCompat
        android:id="@+id/switchRound"
        android:layout_width="0dp"
        android:layout_height="48dp"
        android:layout_marginTop="30dp"
        android:text="@string/round_up_tip"
        app:layout_constraintTop_toBottomOf="@+id/spinner"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintWidth_percent="0.9"/>

    <TextView
        android:id="@+id/tvTotalTip"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:textSize="40sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/switchRound"
        app:layout_constraintWidth_percent="0.9"
        tools:text="@string/tip_amount" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 2. Source Code XML

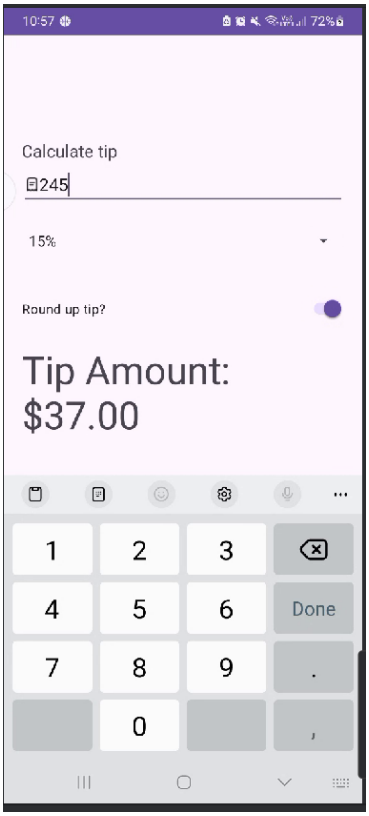
B. Output Program



Gambar 1. Screenshot Hasil Jawaban XML



Gambar 2. Screenshot Hasil Jawaban XML



Gambar 3. Screenshot Hasil Jawaban XML

C. Pembahasan

MainActivity.kt:

```
private lateinit var binding: ActivityMainBinding
```

Menggunakan ViewBinding

Daftar Pilihan Tip

```
private val tipOptions = listOf("10%", "15%", "20%")
```

Daftar preset persentase tip yang bisa dipilih pengguna di Spinner (dropdown menu).

onCreate()

Diatur saat pertama kali activity dijalankan.

Menghubungkan ViewBinding, mengatur adapter Spinner, dan menetapkan listener ke input dan switch.

Spinner listener: Saat pilihan persentase tip berubah, fungsi updateTip() dipanggil.

TextWatcher: Mendeteksi perubahan pada EditText (input tagihan).

Switch listener: Deteksi ketika pengguna ingin membulatkan tip ke atas.

updateTip()

```
val amount = binding.etBill.text.toString().toDoubleOrNull()
?: 0.0
```

Ambil nilai tagihan dari input (jika kosong atau salah, default ke 0).

```
val tipPercent = tipText.removeSuffix("%").toDoubleOrNull() ?:
15.0
```

Ambil persentase tip dari Spinner, hilangkan tanda %, dan ubah ke Double.

```
val roundUp = binding.switchRound.isChecked
```

Cek apakah opsi pembulatan aktif.

```
binding.tvTotalTip.text = getString(R.string.tip_amount, tip)
```

Tampilkan hasil perhitungan tip di TextView.

calculateTip()

```
var tip = tipPercent / 100 * amount
if (roundUp) {
    tip = ceil(tip)
}
```

Hitung nilai tip dari persentase × total.

Jika pembulatan aktif, gunakan ceil() untuk membulatkan ke atas.

```
return NumberFormat.getCurrencyInstance().format(tip)
```

Format tip ke bentuk mata uang lokal (misal: $1.50).

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Andra-Braputra/PrakMobile/tree/main/Modul2/TipXML

COMPOSE

A. Source Code

MainActivity.kt

```
1  package com.example.tipcompose
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.Row
10 import androidx.compose.foundation.layout.Spacer
11 import androidx.compose.foundation.layout.fillMaxSize
12 import androidx.compose.foundation.layout.fillMaxWidth
13 import androidx.compose.foundation.layout.height
14 import androidx.compose.foundation.layout.padding
15 import androidx.compose.foundation.layout.safeDrawingPadding
16 import androidx.compose.foundation.layout.statusBarsPadding
17 import androidx.compose.foundation.rememberScrollState
18 import androidx.compose.foundation.text.KeyboardOptions
19 import androidx.compose.foundation.verticalScroll
20 import androidx.compose.material3.DropdownMenuItem
21 import androidx.compose.material3.ExperimentalMaterial3Api
22 import androidx.compose.material3.ExposedDropdownMenuBox
23 import androidx.compose.material3.ExposedDropdownMenuDefaults
24 import androidx.compose.material3.MaterialTheme
25 import androidx.compose.material3.Surface
26 import androidx.compose.material3.Switch
27 import androidx.compose.material3.Text
28 import androidx.compose.material3.TextField
29 import androidx.compose.runtime.Composable
30 import androidx.compose.runtime.getValue
31 import androidx.compose.runtime.mutableStateOf
32 import androidx.compose.runtime.remember
33 import androidx.compose.runtime.setValue
34 import androidx.compose.ui.Alignment
35 import androidx.compose.ui.Modifier
36 import androidx.compose.ui.res.stringResource
37 import androidx.compose.ui.text.input.KeyboardType
38 import androidx.compose.ui.tooling.preview.Preview
39 import androidx.compose.ui.unit.dp
40 import com.example.tipcompose.ui.theme.TipComposeTheme
41 import java.text.NumberFormat
42 import kotlin.math.ceil
43
44
45 class MainActivity : ComponentActivity() {
46     override fun onCreate(savedInstanceState: Bundle?) {
```

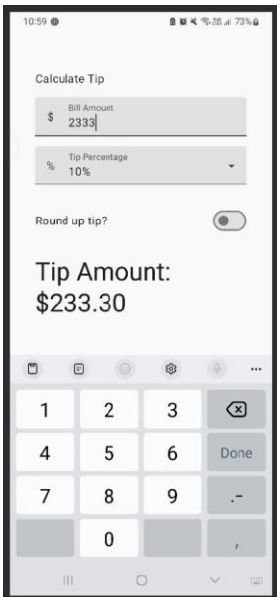
```
47          super.onCreate(savedInstanceState)
48          enableEdgeToEdge()
49          setContent {
50              TipComposeTheme {
51                  Surface(
52                      modifier = Modifier.fillMaxSize(),
53                  ) {
54                      TipTimeLayout()
55                  }
56              }
57          }
58      }
59  }
60
61
62  @Preview(showBackground = true)
63  @Composable
64  fun TipTimeLayoutPreview() {
65      TipComposeTheme {
            TipTimeLayout()
        }
    }
    @OptIn(ExperimentalMaterial3Api::class)
    @Composable
    fun TipTimeLayout() {
        var amountInput by remember { mutableStateOf("") }

        var expanded by remember { mutableStateOf(false) }
        val options = listOf("10%", "15%", "20%")
        var selectedOptionText by remember { mutableStateOf(options[0])
    }

        var roundUp by remember { mutableStateOf(false) }

        val amount = amountInput.toDoubleOrNull() ?: 0.0
        val                          tipPercent                          =
    selectedOptionText.removeSuffix("%").toDoubleOrNull() ?: 15.0
        val tip = calculateTip(amount, tipPercent, roundUp)

        Column(
            modifier = Modifier.fillMaxSize()
                .statusBarsPadding()
                .padding(horizontal = 40.dp)
                .verticalScroll(rememberScrollState())
                .padding(top = 20.dp)
                .safeDrawingPadding(),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Top
        ) {
            Text(
                text = stringResource(R.string.calculate_tip),
                modifier = Modifier
```

```
                .padding(bottom = 16.dp, top = 40.dp)
                .align(alignment = Alignment.Start)
        )

        EditNumberField(
            value = amountInput,
            onValueChange = { amountInput = it },
            modifier = Modifier
                .fillMaxWidth()
                .padding(bottom = 16.dp)
        )

        ExposedDropdownMenuBox(
            expanded = expanded,
            onExpandedChange = { expanded = !expanded },
            modifier = Modifier
                .padding(bottom = 32.dp)
                .fillMaxWidth()
        ) {
            TextField(
                readOnly = true,
                value = selectedOptionText,
                onValueChange = {},
                leadingIcon = { Text(text = "%") },
                label = { Text("Tip Percentage") },
                trailingIcon = {

ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded)
                },
                modifier = Modifier
                    .menuAnchor()
                    .fillMaxWidth(),
                colors                                                  =
ExposedDropdownMenuDefaults.textFieldColors()
            )

            ExposedDropdownMenu(
                expanded = expanded,
                onDismissRequest = { expanded = false }
            ) {
                options.forEach { selectionOption ->
                    DropdownMenuItem(
                        text = { Text(selectionOption) },
                        onClick = {
                            selectedOptionText = selectionOption
                            expanded = false
                        }
                    )
                }
            }
        }
        Row(
```

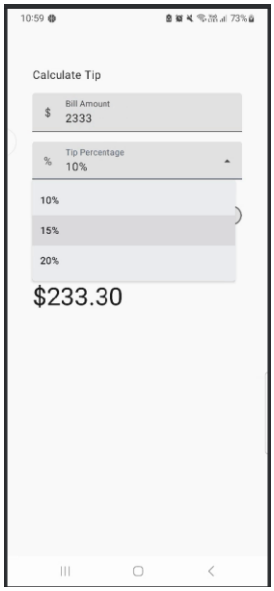
```
            modifier = Modifier
                .fillMaxWidth()
                .padding(bottom = 32.dp),
            verticalAlignment = Alignment.CenterVertically,
        ) {
            Text(modifier = Modifier.weight(1f), text = "Round up
tip?")
            Switch(
                modifier = Modifier,
                checked = roundUp,
                onCheckedChange = { roundUp = it }
            )
        }

        Text(
            text = stringResource(R.string.tip_amount, tip),
            style = MaterialTheme.typography.displaySmall
        )

        Spacer(modifier = Modifier.height(40.dp))
    }
}

@Composable
fun EditNumberField(
    value: String,
    onValueChange: (String) -> Unit,
    modifier: Modifier = Modifier
) {
    TextField(
        value = value,
        onValueChange = onValueChange,
        singleLine = true,
        leadingIcon = { Text(text = "$") },
        label = { Text(stringResource(R.string.bill_amount)) },
        keyboardOptions     =     KeyboardOptions(keyboardType     =
KeyboardType.Number),
        modifier = modifier
    )
}

private fun calculateTip(amount: Double, tipPercent: Double = 15.0,
roundUp: Boolean): String {
    var tip = tipPercent / 100 * amount
    if (roundUp) {
        tip = ceil(tip)
    }
    return NumberFormat.getCurrencyInstance().format(tip)
}
```

Tabel 3. Source Code Compose

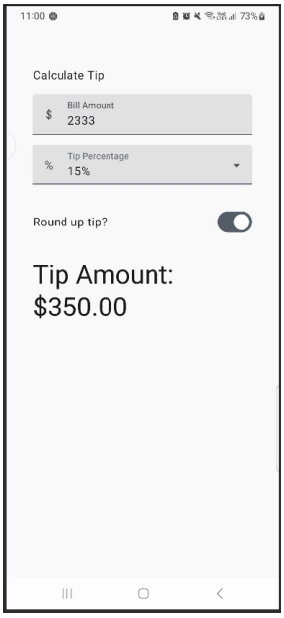
B. Output Program



Gambar 4. Source Code Jawaban Compose



Gambar 5. Source Code Jawaban Compose



Gambar 6. Source Code Jawaban Compose

C. Pembahasan

MainActivity.kt:

1. MainActivity

   Merupakan entry point aplikasi.

   Fungsi setContent digunakan untuk menampilkan UI berbasis Compose.

   TipTimeLayout() adalah fungsi composable yang ditampilkan di layar.

2. TipTimeLayout()

Fungsi ini membangun tampilan utama aplikasi:

State

   amountInput: Input jumlah tagihan (dari pengguna).

   expanded: Status apakah dropdown terbuka atau tidak.

   options: Daftar pilihan persentase tip.

   selectedOptionText: Persentase tip yang dipilih.

   roundUp: Apakah hasil tip akan dibulatkan ke atas.

Perhitungan Tip

amount: Konversi input string ke angka double.

tipPercent: Konversi pilihan string (misalnya "15%") ke angka.

tip: Nilai tip yang dihitung dengan fungsi calculateTip.

UI

Tampilan utama terdiri dari:

Judul: "Calculate Tip"

Input Tagihan: TextField untuk masukkan jumlah tagihan.

Dropdown Tip: Dropdown untuk memilih persentase tip.

Switch Pembulatan: Apakah tip ingin dibulatkan ke atas.

Hasil Tip: Menampilkan hasil perhitungan tip.

3. EditNumberField()

Composable untuk membuat TextField khusus input angka, dengan ikon dolar di depan.

4. calculateTip()

Fungsi untuk menghitung jumlah tip:

Rumus: tip = (persen tip / 100) * jumlah tagihan

Jika roundUp bernilai true, maka hasil akan dibulatkan ke atas dengan ceil.

Hasil dikembalikan dalam format mata uang lokal.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Andra-Braputra/PrakMobile/tree/main/Modul1/TipCompose

SOAL 2

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

A. Jawaban

Jetpack Compose menyatukan logika dan tampilan dalam satu alur kode Kotlin dengan pendekatan deklaratif berbasis state. Perubahan pada data langsung memicu perubahan tampilan tanpa harus memanggil fungsi pembaruan secara manual.

Kelebihan

- Kode Lebih Singkat & Modular

UI ditulis dalam fungsi @Composable, lebih ringkas dan bisa digunakan ulang.

- UI Reaktif & Mudah Diubah

Mengandalkan state → UI langsung berubah tanpa panggil notifyDataSetChanged, dll.

- Integrasi Preview Real-time

Bisa lihat hasil tampilan langsung tanpa compile seluruh app.

Kekurangan

- UI dan logic digabung

Baris kode akan lebih panjang dan membingunkan.

XML dengan ViewBinding masih menggunakan pendekatan imperatif. Dengan ViewBinding, setiap elemen UI di XML langsung direpresentasikan sebagai properti dalam class binding yang dihasilkan secara otomatis, sehingga pengembang dapat mengakses dan memodifikasi komponen UI secara aman dan lebih efisien.

Kelebihan

- Terbiasa & Familiar

Banyak developer Android sudah berpengalaman dengan XML.

- Documented & Stabil

Digunakan selama bertahun-tahun dan dokumentasinya sangat luas.

Kekurangan

- Rigid & Sulit Dikelola

Membuat UI dinamis atau kompleks butuh banyak kode imperative dan nested views.