LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2

Android Layout With Compose

Oleh:

Alya Rosaan NIM. 2310817320006

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 2024

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 : Alya Rosaan
NIM : 2310817320006

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

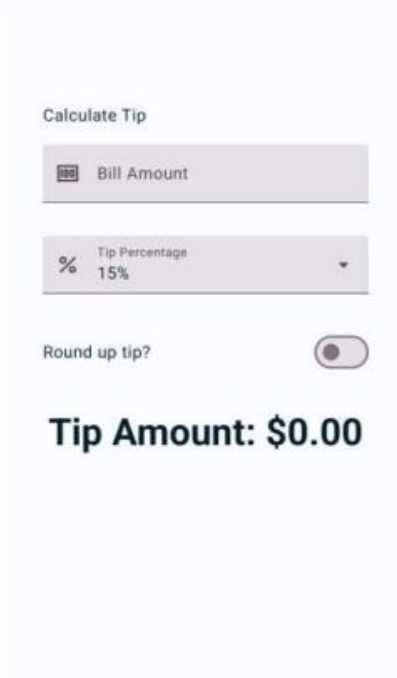
DAFTAR TABEL

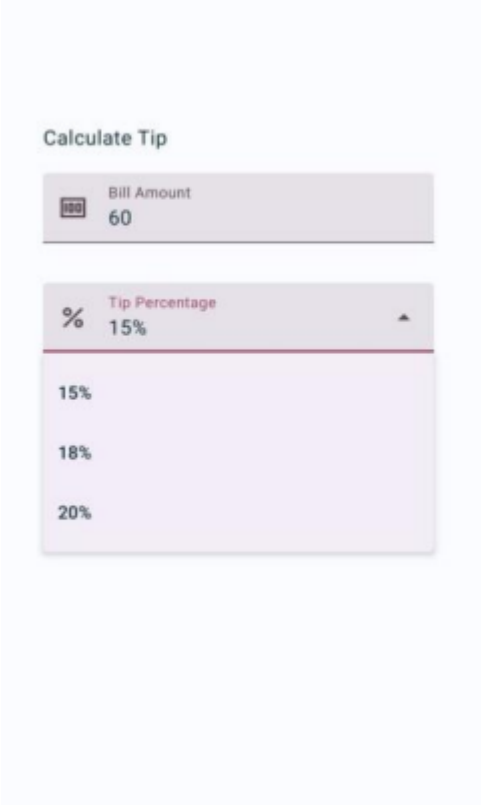
SOAL 1

Soal Praktikum:

1. Buatlah sebuah aplikasi kalkulator tip menggunakan XML dan Jetpack Compose yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:
   a. Input biaya layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
   b. Pilihan persentase tip: Pengguna dapat memilih persentase tip yang diinginkan.
   c. Pengaturan pembulatan tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
   d. Tampilan hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.

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





XML

A. Source Code

MainActivity.kt

```
1   package com.example.mycalculatortipxml
2
3   import android.os.Bundle
4   import android.widget.*
5   import androidx.appcompat.app.AppCompatActivity
6   import kotlin.math.ceil
7
8   class MainActivity : AppCompatActivity() {
9
10      private lateinit var amountInput: EditText
11      private lateinit var tipSpinner: Spinner
12      private lateinit var roundUpSwitch: Switch
13      private lateinit var resultText: TextView
14
15      private val tipOptions = listOf(0.15, 0.18, 0.20)
16
17      override fun onCreate(savedInstanceState: Bundle?) {
18          super.onCreate(savedInstanceState)
19          setContentView(R.layout.activity_main)
20
21          amountInput = findViewById(R.id.billAmountInput)
22          tipSpinner = findViewById(R.id.tipSpinner)
23          roundUpSwitch = findViewById(R.id.roundUpSwitch)
```

```
24          resultText = findViewById(R.id.resultText)
25
26          val tipStrings = tipOptions.map { "${(it * 100).toInt()}%"
27  }
28          val adapter = ArrayAdapter(this,
29  android.R.layout.simple_spinner_dropdown_item, tipStrings)
30          tipSpinner.adapter = adapter
31
32          val calculate = {
33              val amount =
34  amountInput.text.toString().toDoubleOrNull() ?: 0.0
35              var tip = amount *
36  tipOptions[tipSpinner.selectedItemPosition]
37              if (roundUpSwitch.isChecked) tip = ceil(tip)
38              resultText.text = "Tip Amount:
39  $${String.format("%.2f", tip)}"
40          }
41
42          amountInput.setOnEditorActionListener { _, _, _ ->
43              calculate()
44              false
45          }
46
47          tipSpinner.onItemSelectedListener = object :
    AdapterView.OnItemSelectedListener {
48              override fun onItemSelected(parent: AdapterView<*>,
49  view: android.view.View?, position: Int, id: Long) {
                    calculate()
50              }
51
52              override fun onNothingSelected(parent: AdapterView<*>)
53  {}
54          }
55
56          roundUpSwitch.setOnCheckedChangeListener { _, _ ->
57  calculate() }
      }
    }
```

Tabel 1. Source Code Jawaban Soal 1

activity_main.xml

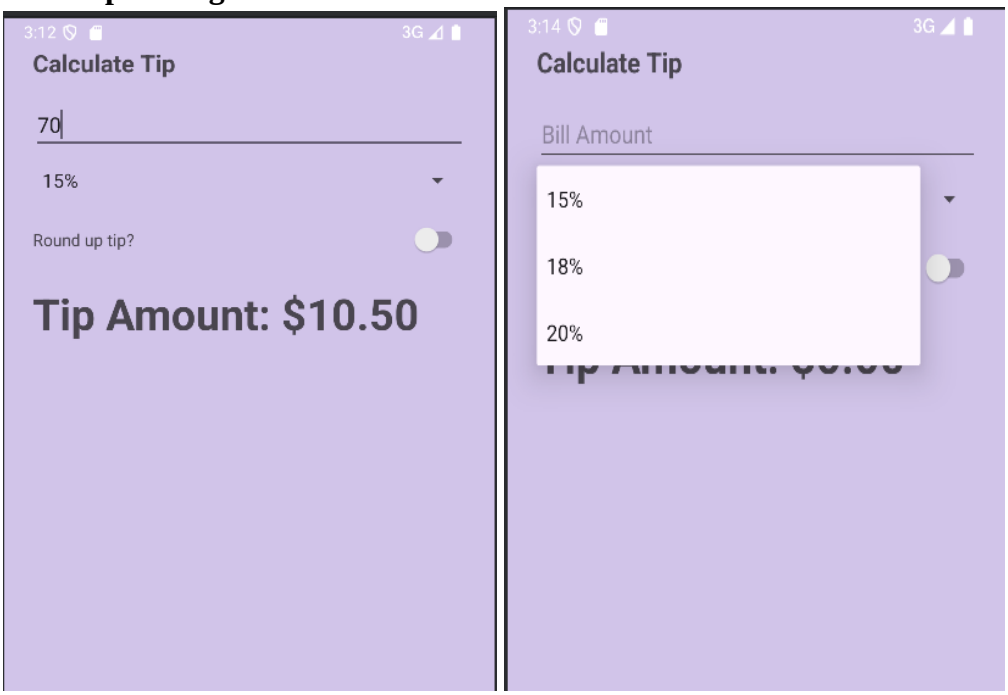
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:id="@+id/mainLayout"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       android:orientation="vertical"
9       android:padding="24dp"
```

```
10      android:background="#D1C4E9">
11
12      <TextView
13          android:id="@+id/titleText"
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content"
16          android:text="Calculate Tip"
17          android:textSize="20sp"
18          android:textStyle="bold" />
19
20      <EditText
21          android:id="@+id/billAmountInput"
22          android:layout_width="match_parent"
23          android:layout_height="wrap_content"
24          android:hint="Bill Amount"
25          android:inputType="numberDecimal"
26          android:layout_marginTop="16dp"/>
27
28
29      <Spinner
30          android:id="@+id/tipSpinner"
31          android:text="Tip Percentage"
32          android:layout_width="match_parent"
33          android:layout_height="wrap_content" />
34
35      <LinearLayout
36          android:layout_width="match_parent"
37          android:layout_height="wrap_content"
38          android:orientation="horizontal"
39          android:layout_marginTop="12dp"
40          android:gravity="center_vertical">
41
42          <TextView
43              android:id="@+id/roundUpLabel"
44              android:layout_width="0dp"
45              android:layout_weight="1"
46              android:layout_height="wrap_content"
47              android:text="Round up tip?" />
48
49          <Switch
50              android:id="@+id/roundUpSwitch"
51              android:layout_width="wrap_content"
52              android:layout_height="wrap_content" />
53      </LinearLayout>
54
55      <TextView
56          android:id="@+id/resultText"
57          android:layout_width="wrap_content"
58          android:layout_height="wrap_content"
59          android:text="Tip Amount: $0.00"
70          android:textSize="36sp"
71          android:textStyle="bold"
```

```
        android:layout_marginTop="24dp"/>
</LinearLayout>
>
```

Tabel 2. Source Code Jawaban Soal 1

B. Output Program



Gambar 1. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

MainActivity.kt:

1. `package com.example.mycalculatortipxml`
   - syntax ini menyatakan bahwa file ini bagian dari package com. Example.mycalculatortipxml
2. ```
   import android.os.Bundle
   import android.widget.*
   import androidx.appcompat.app.AppCompatActivity
   import kotlin.math.ceil
   ```
   - Syntax ini mengimpor kelas2 yang dibutuhkan seperti bundle untuk menyimpan state , kemudian ada edit text, spinner switch textview untuk elemn UI, appcompat activity sebagai superclass dari activity ini, lalu ada ceil dari Kotlin untuk mebulatkan angka keatas
3. `class MainActivity : AppCompatActivity() {`
   - syntax ini merupakan kelas utama dari aplikasi, mewarisi dari appcompatactivity
4. ```
   private lateinit var amountInput: EditText
   private lateinit var tipSpinner: Spinner
   ```

```
private lateinit var roundUpSwitch: Switch
private lateinit var resultText: TextView
```

- baris-ini mendeklarasikan variable UI dan akan di linkan ke elemen di XML lewat findviewById
- private val tipOptions = listOf(0.15, 0.18, 0.20) ini baris kode yang memberi daftar pilihan persen tip: 15%, 18%, 20%

5.
```
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
```

- Fungsi oncreate dipanggil saat actifity dibuat. setContentView menetapkan tampilan activity dari file xmlnya

6.
```
amountInput = findViewById(R.id.billAmountInput)
tipSpinner = findViewById(R.id.tipSpinner)
roundUpSwitch = findViewById(R.id.roundUpSwitch)
resultText = findViewById(R.id.resultText)
```

- sintax ini menghubungkan property Kotlin dengan elemen UI di layout XML

7.
```
val tipStrings = tipOptions.map { "${(it * 100).toInt()}%" }
val adapter = ArrayAdapter(this,
android.R.layout.simple_spinner_dropdown_item, tipStrings)
tipSpinner.adapter = adapter
```

- Mengubah list Double menjadi string seperti "15%".
- Membuat adapter untuk spinner agar menampilkan daftar persentase tip.

8.
```
val calculate = {
    val amount = amountInput.text.toString().toDoubleOrNull()
?: 0.0
    var tip = amount *
tipOptions[tipSpinner.selectedItemPosition]
    if (roundUpSwitch.isChecked) tip = ceil(tip)
    resultText.text = "Tip Amount: $${String.format("%.2f",
tip)}"}
```

- pada baris pertama berfungsi untuk mengambil nilai dari input
- kedua, Hitung tip berdasarkan pilihan spinner
- pada baris if Bulatkan keatas jika switch aktif
- Baris result akan menampilkan hasil ke TextView

9.
```
amountInput.setOnEditorActionListener { _, _, _ ->
    calculate()
    false
```

- baris kode ini akan menjalankan kalkulasi saat user selesai mengetik angka dan meneka tombol aksi seperti enter

10.
```
   tipSpinner.onItemSelectedListener = object :
AdapterView.OnItemSelectedListener{
   override fun onItemSelected(parent: AdapterView<*>, view:
android.view.View?, position: Int, id: Long) {
       calculate()}
 override fun onNothingSelected(parent: AdapterView<*>) {}}
```

- Kode ini mendaftarkan sebuah objek OnItemSelectedListener ke tipSpinner. Jadi, setiap kali user memilih item baru di Spinner, akan terjadi: Fungsi

onItemSelected() dipanggil. Fungsi calculate() dijalankan, yang akan menghitung ulang nilai tip berdasarkan pilihan baru itu.

11. `roundUpSwitch.setOnCheckedChangeListener { _, _ -> calculate() }` pada baris ini calculate dijalankan saat switch diubah(aktf/tidak)

activity_main.xml:

1. `<?xml version="1.0" encoding="utf-8"?>`
   - Baris deklarasi XML standar. Menyatakan versi XML dan encoding file.
2. `<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:app=http://schemas.android.com/apk/res-auto`
   - Membuka tag LinearLayout, yang artinya semua komponen UI akan disusun secara linear (atas ke bawah atau kiri ke kanan).
   - xmlns:android dan xmlns:app adalah deklarasi namespace agar atribut android: dan app: dikenali.
3. `android:id="@+id/mainLayout"`
   - Memberikan ID pada layout utama, agar bisa diakses dari Kotlin pakai findViewById(R.id.mainLayout).
4. `android:layout_width="match_parent" android:layout_height="match_parent"`
   - Mengatur ukuran layout utama agar penuh lebar dan tinggi layar.
5. `android:orientation="vertical"`
   - Menyusun anak-anak LinearLayout secara vertikal (dari atas ke bawah).
6. `android:padding="24dp"`
   - Memberi jarak 24dp dari tepi layout ke isi di dalamnya.
7. `android:background="#D1C4E9">`
   - Memberi warna ungu muda sebagai background layout.
8.
```
<TextView
    android:id="@+id/titleText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Calculate Tip"
    android:textSize="20sp"
   android:textStyle="bold" />
```
   - Kode ini akan menampilakn teks di paling atas yang bertuliskan “Calculate Tip” tampil tebal dan besar
9.
```
<EditText
    android:id="@+id/billAmountInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Bill Amount"
    android:inputType="numberDecimal"
    android:layout_marginTop="16dp"/>
```
   - Kode ini akan menampilakan Kotak input tempat user mengetik jumlah tagihan.
   - hint memberi teks petunjuk "Bill Amount"
   - inputType="numberDecimal" membatasi input hanya angka dan titik desimal.
10.
```
  <Spinner
    android:id="@+id/tipSpinner"
```

```
        android:text="Tip Percentage"
        android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

- Kode ini akan membuat Dropdown yang akan menampilkan pilihan tip (isi list-nya diset di Kotlin).

```
11.     <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="12dp"
        android:gravity="center_vertical">
```

- Membuat baris horizontal berisi label dan switch (untuk rounding tip).
- gravity="center_vertical" memastikan teks dan switch sejajar tengah secara vertikal.

```
12.         <TextView
            android:id="@+id/roundUpLabel"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
          android:text="Round up tip?" />
```

- Label "Round up tip?" di kiri.
- layout_weight="1" membuat label ini mengisi ruang sebanyak mungkin di baris.

```
13.         <Switch
            android:id="@+id/roundUpSwitch"
            android:layout_width="wrap_content"
           android:layout_height="wrap_content" />
```

- Switch yang bisa diaktifkan untuk membulatkan tip ke atas.

```
14.     <TextView
        android:id="@+id/resultText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tip Amount: $0.00"
        android:textSize="36sp"
        android:textStyle="bold"
        android:layout_marginTop="24dp"/>
```

- Baris2 kode ini Menampilkan hasil akhir: "Tip Amount: $xx.xx"
- Teks besar dan bold supaya mencolok.

Jetpack Compose

D. Source Code

MainActivity.kt

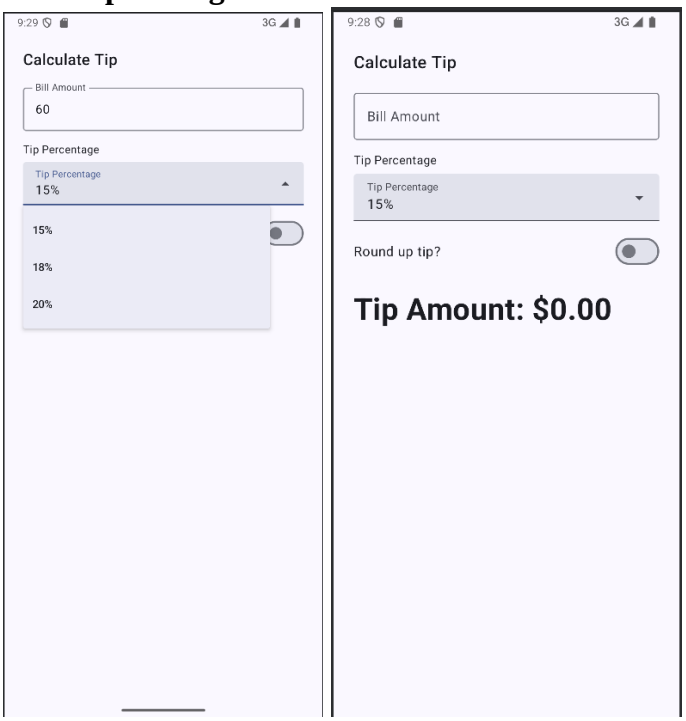
```
1   package com.example.mycalculatortip
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.activity.enableEdgeToEdge
7   import androidx.compose.foundation.layout.*
```

```
8  import androidx.compose.foundation.text.KeyboardOptions
9  import androidx.compose.material3.*
10 import androidx.compose.runtime.*
11 import androidx.compose.ui.Alignment
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.text.input.KeyboardType
14 import androidx.compose.ui.text.font.FontWeight
15 import androidx.compose.ui.unit.dp
16 import androidx.compose.ui.unit.sp
17 import com.example.mycalculatortip.ui.theme.MyCalculatorTipTheme
18 import kotlin.math.ceil
19
20 class MainActivity : ComponentActivity() {
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         enableEdgeToEdge()
24         setContent {
25             MyCalculatorTipTheme {
26                 Scaffold(modifier = Modifier.fillMaxSize()) {
   innerPadding ->
27                     TipCalculatorApp(modifier =
   Modifier.padding(innerPadding))
28                 }
29             }
30         }
31     }
32 }
33
34 @Composable
35 fun TipCalculatorApp(modifier: Modifier = Modifier) {
36     var amountInput by remember { mutableStateOf("") }
37     var selectedTip by remember { mutableStateOf(0.15) }
38     var roundUp by remember { mutableStateOf(false) }
39
40     val amount = amountInput.toDoubleOrNull() ?: 0.0
41     var tip = amount * selectedTip
42     if (roundUp) tip = ceil(tip)
43
44     Column(
45         modifier = modifier
46             .fillMaxSize()
47             .padding(24.dp),
48     ) {
49         Text(
50             text = "Calculate Tip",
51             fontSize = 20.sp,
52             fontWeight = FontWeight.Medium
53         )
54
55         Spacer(modifier = Modifier.height(16.dp))
56
57         OutlinedTextField(
```

```
58              value = amountInput,
59              onValueChange = { amountInput = it },
60              label = { Text("Bill Amount") },
61              modifier = Modifier.fillMaxWidth(),
62              keyboardOptions = KeyboardOptions(keyboardType =
    KeyboardType.Number)
63
64          )
65
66          Spacer(modifier = Modifier.height(12.dp))
67          Text("Tip Percentage", fontSize = 14.sp, modifier =
    Modifier.padding(bottom = 4.dp))
68
69          DropdownMenuTip(selectedTip) { selectedTip = it }
70
71          Spacer(modifier = Modifier.height(12.dp))
72
            Row(
73              verticalAlignment = Alignment.CenterVertically,
74              modifier = Modifier.fillMaxWidth()
75          ) {
76              Text("Round up tip?", modifier = Modifier.weight(1f))
77              Switch(checked = roundUp, onCheckedChange = { roundUp
    = it })
78          }
79
80          Spacer(modifier = Modifier.height(24.dp))
82
83          Text(
84              text = "Tip Amount: $${String.format("%.2f", tip)}",
85              fontSize = 36.sp,
86              fontWeight = FontWeight.Bold
87          )
88      }
89  }
90  @OptIn(ExperimentalMaterial3Api::class)
91  @Composable
92  fun DropdownMenuTip(selectedTip: Double, onTipSelected: (Double)
93  -> Unit) {
94      val tips = listOf(0.15, 0.18, 0.20)
95      var expanded by remember { mutableStateOf(false) }
96      var selectedOptionText by remember {
97          mutableStateOf("${(selectedTip * 100).toInt()}%")
98      }
99
100     ExposedDropdownMenuBox(
101         expanded = expanded,
102         onExpandedChange = { expanded = !expanded }
103     ) {
104         TextField(
105             readOnly = true,
106             value = selectedOptionText,
```

```
107             onValueChange = { },
108             label = { Text("Tip Percentage") },
109             trailingIcon = {
110                 ExposedDropdownMenuDefaults.TrailingIcon(expanded
    = expanded)
111             },
112             modifier = Modifier
123                 .menuAnchor()
124                 .fillMaxWidth(),
125             colors =
    ExposedDropdownMenuDefaults.textFieldColors()
126         )
127 
128         ExposedDropdownMenu(
129             expanded = expanded,
130             onDismissRequest = { expanded = false }
131         ) {
132             tips.forEach { tip ->
133                 DropdownMenuItem(
134                     text = { Text("${(tip * 100).toInt()}%") },
135                     onClick = {
136                         selectedOptionText = "${(tip *
    100).toInt()}%"
137                         onTipSelected(tip)
138                         expanded = false
139                     }
140                 )
141             }
142         }
    }
}
```

E. Output Program



F. Pembahasan

MainActivity.kt:

1. `package com.example.mycalculatortip`
   - Menentukan package aplikasi.
2. ```
   import android.os.Bundle
   import androidx.activity.ComponentActivity
   import androidx.activity.compose.setContent
   import androidx.activity.enableEdgeToEdge
   ```
   ….
   - Mengimpor semua komponen Compose dan Kotlin yang dibutuhkan, seperti Text, Column, OutlinedTextField, Switch, dan lain-lain.
3. ```
   class MainActivity : ComponentActivity() {
       override fun onCreate(savedInstanceState: Bundle?) {
           super.onCreate(savedInstanceState)
           enableEdgeToEdge()
   ```
   - MainActivity adalah titik awal aplikasi. enableEdgeToEdge() membuat konten bisa tampil hingga ke tepi layar.
4. `MyCalculatorTipTheme {`
   - Menentukan konten UI menggunakan Compose dan tema custom MyCalculatorTipTheme.
5. `Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding -> TipCalculatorApp(modifier = Modifier.padding(innerPadding))`
   - Scaffold adalah layout dasar dari Material 3. Di dalamnya, fungsi TipCalculatorApp() dipanggil sebagai isi utama aplikasi.
6. ```
   @Composable
   fun TipCalculatorApp(modifier: Modifier = Modifier) {
   ```
   ```

- fungsi compose utama

7. ```
   var amountInput by remember { mutableStateOf("") }
       var selectedTip by remember { mutableStateOf(0.15) }
       var roundUp by remember { mutableStateOf(false) }
   ```
   - Remember digunakan untuk menyimpan nilai input pengguna seperti: amount input yaitu string dari input tagihan, selected tip untuk nilai persen tip, dan round up apakah tip dibulatkan atau tidak
8. ```
   val amount = amountInput.toDoubleOrNull() ?: 0.0
       var tip = amount * selectedTip
       if (roundUp) tip = ceil(tip)
   ```
   - baris kode2 ini mengonversi input ke double, lalu menghitung nilai tip. Jika Roundup aktif, tip akan dibulatkan keatas
9. ```
   Column(
       modifier = modifier
         .fillMaxSize()
         .padding(24.dp),
   ```
   - Baris untuk tampilan ui, layout dibuat vertical, semua komponen ui berada di dalam column
10. ```
    Text(
          text = "Calculate Tip",
          fontSize = 20.sp,
          fontWeight = FontWeight.Medium
    ```
    - Baris kode2 ini mendekalarsikan “text” sebagaia variable yang di isi dengan judul apilikasi dengan ukuran 20, dan bobot fontnya tebal sedang
11. ```
    Spacer(modifier = Modifier.height(12.dp))
    ```
    - Kode ini Memberi jarak antar komponen.
12. ```
    OutlinedTextField(
     value = amountInput,
     onValueChange = { amountInput = it },
     label = { Text("Bill Amount") },
     modifier = Modifier.fillMaxWidth(),
     keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number)
    ```
    - baris kode ii memberkan Input field untuk jumlah tagihan. OutlinedTextField memberi garis luar.
13. ```
    Spacer(modifier = Modifier.height(12.dp))
            Text("Tip Percentage", fontSize = 14.sp, modifier =
    Modifier.padding(bottom = 4.dp))
            DropdownMenuTip(selectedTip) { selectedTip = it }
    ```
    - Baris-baris kode ini membuat Label dan dropdown untuk memilih persen tip. Pemilihan akan mengubah selectedTip.
14. ```
    Row(
       verticalAlignment = Alignment.CenterVertically,
       modifier = Modifier.fillMaxWidth()) {
       Text("Round up tip?", modifier = Modifier.weight(1f))
       Switch(checked = roundUp, onCheckedChange = { roundUp = it })
    ```
    - Baris kode ini membuat baris horizontal yang berisi label dan switch untuk membulatkan tip, lalu modifier.weight untuk membuat teks mengisi ruang yang tersisa

15.
```
Text(
 text = "Tip Amount: $${String.format("%.2f", tip)}",
 fontSize = 36.sp,
 fontWeight = FontWeight.Bold
```
- baris kod ini membaut variable text berisi tip amount, Menampilkan hasil perhitungan tip dengan format 2 angka desimal. Beukuran 26 dan tebal

16.
```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DropdownMenuTip(selectedTip: Double, onTipSelected: (Double) ->
Unit) {
```
- buat Fungsi terpisah untuk membuat dropdown menu custom.

17.
```
val tips = listOf(0.15, 0.18, 0.20)
    var expanded by remember { mutableStateOf(false) }
    var selectedOptionText by remember {
        mutableStateOf("${(selectedTip * 100).toInt()}%")}
```
- var tips berisi daftar pilihan, var expanded untuk kontrol tampilan menu, dan var selectedOptionText menampilkan pilihan saat ini.

18.
```
ExposedDropdownMenuBox(
  expanded = expanded,
  onExpandedChange = { expanded = !expanded }
```
- ExposedDropdownMenuBox diambil dari Material3 untuk membuat dropdown interaktif.

19.
```
TextField(
  readOnly = true,
  value = selectedOptionText,
  onValueChange = { },
   label = { Text("Tip Percentage") },
   trailingIcon = {
  ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded)
```
- Menampilkan teks dropdown sebagai TextField yang tidak bisa diketik. Read only bernilai true

20.
```
ExposedDropdownMenu(
    expanded = expanded,
    onDismissRequest = { expanded = false }
```
- ExposedDropdownMenu adalah komponen dropdown menu yang muncul di bawah TextField dari ExposedDropdownMenuBox.
- expanded: Boolean yang menentukan apakah menu ditampilkan atau tidak.
- onDismissRequest: Fungsi yang dipanggil ketika pengguna meklik di luar dropdown atau menutup dropdown, dan kita set expanded = false untuk menyembunyikannya.

21.
```
tips.forEach { tip ->
```
- Melakukan iterasi untuk setiap nilai tip di daftar tips (yaitu 0.15, 0.18, 0.20).

22.
```
text = { Text("${(tip * 100).toInt()}%") },
  onClick = {selectedOptionText = "${(tip *
100).toInt()}%"onTipSelected(tip)
 expanded = false
```
- Variabel text untuk menampilkan teks misalnya "15%", "18%", "20%" dengan mengonversi nilai tip ke persen.
- onClick: Ketika salah satu item diklik, selectedOptionText diperbarui agar tampilan TextField juga ikut berubah lalu onTipSelected(tip) memanggil callback untuk

memberi tahu komponen utama nilai tip baru. Terakhir expanded = false menutup dropdown menu setelah memilih.

2. Jelaskan perbedaan dari implementasi XML dan Jetpack Compose beserta kelebihan dan kekurangan dari masing-masing implementasi
   Jawaban: perbedaan utama antara implementasu UI dengan XML dan jetpack Compose terletak pada pendekatan dan Bahasa yang digunakan. XML menggunakan file berbasis teks untuk mendefinisikan tata letak, sedangkan Compose menggunakan fungsi Kotlin untuk mendefinisikan UI secara deklaratif. XML lebih tradisional dan bersifat imperatif, sedangkan Compose lebih modern, deklaratif, dan reaktif.
   Keunggulan XML Mudah untuk menggabungkan Compose dengan tampilan yang sudah ada (View). XML mempertahankan kecepatan build yang lebih baik dibandingkan Compose, meskipun performa runtime sebanding.
   Sedangkan kekurangannya : Membutuhkan banyak kode untuk menangani perubahan dinamis dalam UI, dan Menggunakan banyak sumber daya untuk mendeteksi perubahan UI.
   Lalu jika Keunggulan Jetpack Compose yaitu kode lebih sedikit lebih ringkas dan mudah dibaca, lebih dinamis, Lebih efisien dalam mendeteksi dan memperbarui perubahan UI dan sederhana
   Kekurangannya beberapa komponen UI mungkin membutuhkan pendekatan yang berbeda dibandingkan dengan XML.
   Kesimpulannya Jika kita mencari pendekatan yang lebih tradisional, familiar, dan cepat, XML masih menjadi pilihan yang baik. Namun, jika kita ingin memanfaatkan pendekatan UI modern, deklaratif, reaktif, dan efisien, Jetpack Compose adalah pilihan yang lebih baik. kita juga dapat menggabungkan kedua pendekatan ini untuk menciptakan aplikasi yang lebih fleksibel.

G. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/ALYAROSAAN/Pemrograman-Mobile-Modul2