LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2

ANDROID LAYOUT WITH COMPOSE

Oleh:

Devi Hafida Ariyani NIM. 2310817220018

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 Praktikum ini dikerjakan oleh:

Nama Praktikan : Devi Hafida Ariyani
NIM : 2310817220018

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Soal Praktikum:

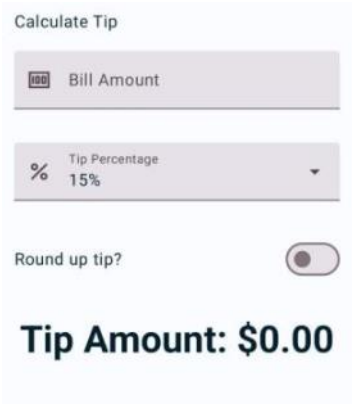
Buatlah sebuah aplikasi kalkulator tip menggunakan XML dan Jetpack Compose yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

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.

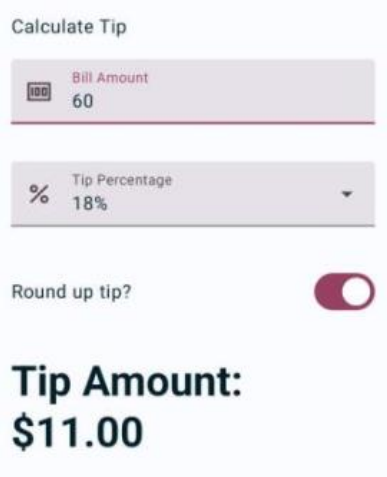
c. Pengaturan pembulatan tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.

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

Tabel 1. Source Code Jawaban Soal 1

```
package com.example.tipcalculator

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Percent
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.tipcalculation.ui.theme.TipCalculationTheme
import kotlin.math.ceil
import     androidx.compose.material.icons.filled.AttachMoney     as
AttachMoney1

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            TipCalculationTheme {
                Surface(modifier  =  Modifier.fillMaxSize(),  color  =
MaterialTheme.colorScheme.background) {
                    TipCalculatorApp()
```

```
                }
            }
        }
    }
}

@Composable
fun TipCalculatorApp() {
    var cost by remember { mutableStateOf("") }
    var tipPercent by remember { mutableStateOf(15) }
    var roundUp by remember { mutableStateOf(false) }

    // Ensure cost is a valid number
    val tip = calculateTip(cost.toDoubleOrNull() ?: 0.0, tipPercent,
roundUp)

    Column(
        modifier = Modifier
            .padding(24.dp)
            .fillMaxWidth(),
        verticalArrangement = Arrangement.spacedBy(20.dp)
    ) {
        Text(
            text = "Calculate Tip",
            style = MaterialTheme.typography.titleMedium,
            modifier = Modifier.padding(bottom = 8.dp)
        )

        OutlinedTextField(
            value = cost,
            onValueChange = { cost = it },
            label = { Text("Bill Amount") },
            leadingIcon    =    {    Icon(Icons.Filled.AttachMoney1,
contentDescription = null) },
            keyboardOptions    =    KeyboardOptions(keyboardType    =
KeyboardType.Number),
            singleLine = true,
            modifier = Modifier.fillMaxWidth()
        )

        TipPercentageDropdown(selected = tipPercent, onSelected = {
tipPercent = it })

        Row(
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.SpaceBetween
        ) {
            Text("Round up tip?")
            Switch(checked = roundUp, onCheckedChange = { roundUp = it
})
        }

        Text(
            text = "Tip Amount: $${String.format("%.2f", tip)}",
            fontSize = 24.sp,
            fontWeight = FontWeight.Bold,
            color = Color(0xFF111111)
```

```
        )
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TipPercentageDropdown(selected: Int, onSelected: (Int) -> Unit) {
    val options = listOf(15, 18, 20)
    var expanded by remember { mutableStateOf(false) }

    ExposedDropdownMenuBox(
        expanded = expanded,
        onExpandedChange = { expanded = !expanded }
    ) {
        OutlinedTextField(
            value = "$selected%",
            onValueChange = {},
            readOnly = true,
            label = { Text("Tip Percentage") },
            leadingIcon      =      {      Icon(Icons.Filled.Percent,
contentDescription = null) },
            trailingIcon                        =                       {
ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded) },
            modifier = Modifier
                .menuAnchor()
                .fillMaxWidth()
        )
        ExposedDropdownMenu(
            expanded = expanded,
            onDismissRequest = { expanded = false }
        ) {
            options.forEach { percent ->
                DropdownMenuItem(
                    text = { Text("$percent%") },
                    onClick = {
                        onSelected(percent)
                        expanded = false
                    }
                )
            }
        }
    }
}

fun calculateTip(amount: Double, tipPercent: Int, roundUp: Boolean):
Double {
    var tip = amount * tipPercent / 100
    if (roundUp) tip = ceil(tip)
    return tip
}
```

activity_main.xml

Tabel 2. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="24dp">

        <TextView
            android:id="@+id/header"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Calculate Tip"
            android:textSize="20sp"
            android:textStyle="bold"
            android:layout_gravity="center_horizontal"
            android:paddingBottom="12dp" />

        <EditText
            android:id="@+id/cost_of_service"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Bill Amount"
            android:inputType="numberDecimal"
            android:drawableLeft="@drawable/ic_attach_money"
            android:drawablePadding="8dp"
            android:minHeight="48dp"
            android:padding="16dp"
            android:background="@drawable/bg_outline"/>

        <Spinner
            android:id="@+id/tip_options"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:minHeight="48dp"
            android:layout_marginTop="16dp" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="space_between"
            android:layout_marginTop="16dp"
            android:minHeight="48dp"
            android:paddingVertical="8dp">

            <TextView
                android:layout_width="0dp"
                android:layout_weight="1"
```

```
                android:layout_height="wrap_content"
                android:text="Round up tip?" />

            <Switch
                android:id="@+id/round_up_switch"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
            android:minHeight="48dp"
            android:padding="16dp"
            android:layout_gravity="center_vertical" />

        </LinearLayout>

        <Button
            android:id="@+id/calculate_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Calculate"
            android:minHeight="48dp"
            android:paddingVertical="12dp"
            android:layout_marginTop="24dp" />

        <TextView
            android:id="@+id/tip_result"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tip Amount: $0.00"
            android:textSize="20sp"
            android:textStyle="bold"
            android:layout_marginTop="32dp" />
    </LinearLayout>
</ScrollView>
```

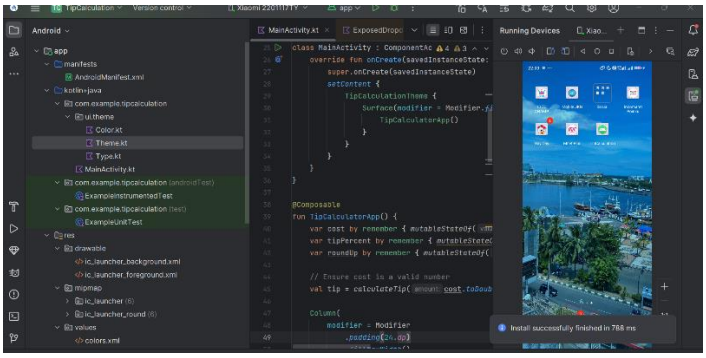
MainActivity.kt

Tabel 3. Source Code Jawaban Soal 1

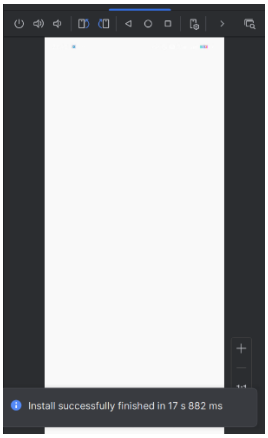
```
package com.example.tipcalculator

import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import com.example.tipcalculation.R
import kotlin.math.ceil

class MainActivity : AppCompatActivity() {

    private lateinit var costOfService: EditText
    private lateinit var tipOptions: Spinner
    private lateinit var roundUpSwitch: Switch
    private lateinit var tipResult: TextView
    private lateinit var calculateButton: Button

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

```
        // Inisialisasi view dari layout
        costOfService = findViewById(R.id.cost_of_service)
        tipOptions = findViewById(R.id.tip_options)
        roundUpSwitch = findViewById(R.id.round_up_switch)
        tipResult = findViewById(R.id.tip_result)
        calculateButton = findViewById(R.id.calculate_button)

        // Set data ke Spinner
        val tipPercentages = arrayOf("15%", "18%", "20%")
        val            adapter            =            ArrayAdapter(this,
android.R.layout.simple_spinner_item, tipPercentages)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdow
n_item)
        tipOptions.adapter = adapter

        // Handle button click
        calculateButton.setOnClickListener {
            calculateTip()
        }
    }

    private fun calculateTip() {
        val costInput = costOfService.text.toString()
        val cost = costInput.toDoubleOrNull()

        // Validasi input
        if (cost == null || cost <= 0) {
            tipResult.text = "Masukkan nominal yang valid"
            return
        }

        // Ambil persentase tip dari pilihan spinner
        val tipPercentage = when (tipOptions.selectedItem.toString()) {
            "20%" -> 0.20
            "18%" -> 0.18
            "15%" -> 0.15
            else -> 0.15 // default fallback
        }

        // Hitung tip
        var tip = cost * tipPercentage
        if (roundUpSwitch.isChecked) {
            tip = ceil(tip)
        }

        // Tampilkan hasil
        tipResult.text = "Tip: Rp %.0f".format(tip)
    }
}

private fun Any.onCreate(savedInstanceState: Bundle?) {

}
```

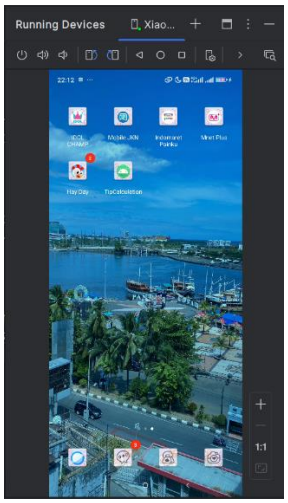
B. Output Program



Gambar 4. Soal 1



Gambar 5. Soal 1



Gambar 6. Soal 1

Gambar 7. Soal 1

C. PEMBAHASAN

MainActivity.kt:

Aplikasi ini adalah kalkulator tip sederhana yang dibuat menggunakan Jetpack Compose dan tema Material3. Saat dijalankan, pengguna akan melihat tampilan untuk memasukkan jumlah tagihan (bill amount), memilih persentase tip (15%, 18%, atau 20%), dan memilih apakah ingin membulatkan jumlah tip atau tidak. Hasil perhitungan tip akan langsung ditampilkan secara dinamis di bawahnya.

Aplikasi dimulai di MainActivity, yang merupakan turunan dari ComponentActivity. Di dalam fungsi onCreate(), composable utama ditampilkan menggunakan setContent {}. Tampilan ini dibungkus dengan TipCalculationTheme untuk menerapkan tema, lalu dilapisi oleh Surface sebagai latar belakang. Di dalamnya terdapat TipCalculatorApp(), yaitu composable utama yang berisi seluruh tampilan antarmuka pengguna.

Fungsi TipCalculatorApp() menggunakan beberapa state Compose untuk menyimpan nilai input dan pilihan pengguna. cost menyimpan jumlah tagihan yang diketik, tipPercent menyimpan persentase tip yang dipilih, dan roundUp menyimpan apakah user ingin membulatkan hasil tip atau tidak. Berdasarkan input tersebut, aplikasi akan memanggil fungsi calculateTip(), yang menghitung jumlah tip berdasarkan persentase dan pilihan pembulatan.

Antarmuka pengguna terdiri dari beberapa komponen: OutlinedTextField untuk input tagihan dengan ikon uang, TipPercentageDropdown sebagai dropdown untuk memilih persentase tip, Switch untuk memilih pembulatan, dan Text untuk menampilkan hasil tip yang sudah dihitung. Semua komponen ini ditata dengan Column agar tampil secara vertikal, dan diberi padding agar tampil rapi.

Komponen TipPercentageDropdown dibuat terpisah sebagai composable custom, menggunakan ExposedDropdownMenuBox dari Material3. Ini memberikan pengalaman dropdown modern dengan tampilan profesional. Saat persentase dipilih, nilainya dikirim kembali ke composable utama menggunakan parameter onSelected.

activity_main.xml:

Kode XML yang diberikan merupakan layout untuk sebuah aplikasi Android yang memiliki antarmuka pengguna (UI) untuk menghitung tip. Layout ini menggunakan ScrollView untuk memastikan elemen-elemen UI dapat digulir jika ruang layar tidak cukup untuk menampilkan semua konten. Di dalam ScrollView, ada LinearLayout yang diatur secara vertikal dan berisi beberapa elemen UI, seperti TextView, EditText, Spinner, Switch, dan Button.

Pada bagian paling atas, terdapat TextView dengan ID header yang menampilkan teks "Calculate Tip". Teks ini berada di tengah secara horizontal dengan ukuran font 20sp dan gaya tebal (bold), diikuti dengan sedikit ruang di bawahnya (android:paddingBottom="12dp"). Ini memberikan judul yang jelas untuk pengguna.

Berikutnya, terdapat EditText dengan ID cost_of_service, yang memungkinkan pengguna untuk memasukkan jumlah tagihan yang ingin dihitung tip-nya. Elemen ini memiliki input type numberDecimal, yang mengarahkan pengguna untuk memasukkan angka desimal. Di sebelah kiri, terdapat gambar dengan atribut android:drawableLeft yang menggunakan ikon uang (ic_attach_money), memberikan konteks visual untuk input jumlah uang. Di bawahnya, ada Spinner dengan ID tip_options, yang menyediakan pilihan persentase tip bagi pengguna, namun masih kosong dan harus diisi dengan data di kode Java.

Ada juga LinearLayout yang mengandung TextView dan Switch untuk fitur "Round up tip?". TextView menjelaskan fungsi dari Switch, dan Switch itu sendiri memungkinkan pengguna untuk memilih apakah mereka ingin membulatkan jumlah tip ke atas atau tidak. Penempatan elemen-elemen ini dalam LinearLayout horizontal dengan atribut gravity="space_between" memastikan bahwa elemen-elemen ini terpisah dengan baik, membuat tampilan lebih rapi.

Bagian bawah layout berisi tombol Button yang memungkinkan pengguna untuk menghitung tip berdasarkan input yang telah dimasukkan. Tombol ini menampilkan teks "Calculate" dan memiliki sedikit ruang di atasnya (android:layout_marginTop="24dp"), memberikan ruang agar elemen-elemen di atasnya tidak terlalu rapat. Di bagian akhir, terdapat TextView dengan ID tip_result, yang akan menampilkan hasil perhitungan tip dalam format teks dengan gaya tebal dan ukuran font 20sp. Ini memberikan umpan balik langsung kepada pengguna setelah mereka menghitung tip.

MainActivity.kt:

Aplikasi ini memungkinkan pengguna untuk menghitung jumlah tip berdasarkan biaya layanan yang dimasukkan dan persentase tip yang dipilih. Di dalam metode onCreate, tampilan antarmuka aplikasi diinisialisasi dengan menghubungkan elemen-elemen UI, seperti EditText untuk biaya layanan, Spinner untuk memilih persentase tip, Switch untuk memilih apakah akan membulatkan tip, dan Button untuk menghitung tip. Semua elemen UI ini dihubungkan dengan variabel di dalam kode menggunakan findViewById().

Pada bagian Spinner, aplikasi menampilkan pilihan persentase tip (15%, 18%, 20%) dengan menggunakan ArrayAdapter. Ketika tombol "Calculate" diklik, aplikasi akan mengambil input dari pengguna, melakukan perhitungan tip berdasarkan persentase yang dipilih, dan

menampilkan hasilnya di TextView. Jika pengguna memilih untuk membulatkan tip, fungsi ceil() digunakan untuk membulatkan hasil perhitungan ke angka terdekat. Input yang tidak valid, seperti biaya yang tidak valid atau kosong, akan memunculkan pesan kesalahan kepada pengguna.

D. TAUTAN GIT

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

SOAL 2

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

1. XML (Traditional View-based UI)

XML adalah cara tradisional untuk mendesain antarmuka pengguna di Android. UI didefinisikan menggunakan file XML yang diproses oleh sistem Android untuk membuat tampilan aplikasi.

Cara Kerja:

- Layout UI dibuat di dalam file XML yang terpisah (misalnya activity_main.xml).
- Elemen-elemen UI seperti TextView, EditText, Button, dan Spinner ditentukan dengan tag XML, dan kemudian dikaitkan dengan kode Kotlin atau Java menggunakan findViewById().

Kelebihan:

- Deklaratif dan Terpisah: Penggunaan XML memisahkan logika aplikasi (kode Kotlin) dan antarmuka pengguna (UI), membuat kode lebih terstruktur dan mudah dipahami.
- Pemakaian Library Pihak Ketiga: XML lebih mudah digunakan bersama dengan berbagai library UI pihak ketiga, yang banyak tersedia di Android.
- Kompatibilitas yang Lebih Luas: XML adalah standar yang sudah ada sejak lama, sehingga hampir semua perangkat Android mendukungnya.

Kekurangan:

- Verbosity: Menggunakan XML bisa memerlukan banyak baris kode, terutama ketika UI menjadi kompleks. Menambahkan banyak elemen bisa membuat kode tampak berantakan.
- Pengelolaan Proyek yang Lebih Sulit: Pengelolaan antarmuka pengguna (UI) yang melibatkan banyak file XML dan pengaturan findViewById() bisa menjadi lebih sulit, terutama ketika aplikasi berkembang.
- Kurang Fleksibel: Membuat perubahan dinamis pada UI (seperti mengganti tampilan berdasarkan data) bisa lebih rumit dan memerlukan pembaruan elemen UI satu per satu.

2. Jetpack Compose (Declarative UI Framework)

Jetpack Compose adalah framework UI deklaratif modern untuk Android, yang memungkinkan pengembang membuat antarmuka pengguna dengan kode Kotlin, tanpa memerlukan XML. Compose memungkinkan pembuatan UI dengan cara yang lebih langsung dan reaktif.

Cara Kerja:

- UI didefinisikan langsung di dalam kode Kotlin menggunakan fungsi-fungsi deklaratif (misalnya Text(), Button(), Column(), dll.).
- Pengguna tidak perlu lagi menggunakan findViewById(), karena elemen-elemen UI langsung dipetakan ke komponen Kotlin.
- UI dapat diperbarui secara reaktif berdasarkan perubahan data menggunakan state management.

Kelebihan:

- Deklaratif dan Mudah Dibaca: Karena Compose menggunakan paradigma deklaratif, UI didefinisikan dalam satu tempat, membuatnya lebih mudah untuk dipahami dan diubah.
- Pengelolaan State yang Lebih Baik: Compose memungkinkan pengelolaan status UI secara lebih efisien menggunakan state dan remember, memungkinkan aplikasi bereaksi terhadap perubahan data secara otomatis.
- Fleksibilitas dan Kinerja: Compose memungkinkan penyesuaian UI dengan lebih bebas, dan lebih efisien dalam hal kinerja dibandingkan dengan XML, terutama untuk animasi dan pembaruan tampilan yang dinamis.
- Integrasi dengan Kotlin: Compose sepenuhnya terintegrasi dengan Kotlin, sehingga memudahkan pengembangan dan pemeliharaan aplikasi.

Kekurangan:

- Kurva Pembelajaran: Meskipun Compose lebih intuitif, ada kurva pembelajaran yang lebih curam bagi pengembang yang belum terbiasa dengan pendekatan deklaratif.
- Keterbatasan Kompatibilitas dengan Library Lama: Karena Jetpack Compose adalah framework baru, beberapa library pihak ketiga atau elemen UI yang lebih tua mungkin belum sepenuhnya kompatibel dengan Compose.
- Masih dalam Pengembangan: Meskipun Jetpack Compose sudah stabil, beberapa fitur dan optimasi masih terus berkembang, yang mungkin memengaruhi keputusan untuk mengadopsinya dalam aplikasi besar yang sudah ada.

Perbandingan dalam Konteks Aplikasi Tip Calculation

- Dengan XML:
    - Pendekatan: UI aplikasi kalkulator tip (misalnya, EditText untuk biaya, Spinner untuk persentase tip, dan Button untuk menghitung) didefinisikan di file XML, dan interaksi dengan elemen-elemen UI dilakukan dalam kode Kotlin menggunakan findViewById().
    - Kelebihan: Lebih mudah digunakan bagi pengembang yang sudah berpengalaman dengan Android SDK tradisional dan XML. Penggunaan library pihak ketiga yang lebih banyak tersedia.

    - Kekurangan: UI menjadi lebih sulit untuk dikelola seiring dengan berkembangnya aplikasi. Perubahan dinamis di UI (misalnya pembaruan hasil kalkulasi) memerlukan manipulasi manual elemen UI.
- Dengan Jetpack Compose:
    - Pendekatan: UI aplikasi kalkulator tip bisa langsung didefinisikan dalam kode Kotlin, menggunakan elemen seperti Text(), Button(), dan Slider(). Dengan Compose, pembaruan hasil tip dapat dilakukan lebih efisien dengan mengubah nilai state dan secara otomatis memperbarui tampilan.
    - Kelebihan: UI lebih fleksibel dan reaktif. Pengelolaan status (seperti hasil kalkulasi tip) lebih mudah dilakukan, dan UI dapat diperbarui secara otomatis saat data berubah.
    - Kekurangan: Penggunaan Jetpack Compose memerlukan waktu belajar dan penyesuaian, terutama bagi pengembang yang belum terbiasa dengan pendekatan deklaratif. Keterbatasan kompatibilitas dengan beberapa library pihak ketiga bisa menjadi tantangan.