

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2**



ANDROID LAYOUT

OLEH:

ZAHRA NABILA

NIM 2310817320007

**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 ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Zahra Nabila
NIM : 2310817320007

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Zulfa Auliya Akbar
NIM. 2210817210026

Muti`a Maulida S.Kom M.T.I
NIP. 19881027 201903 20 13

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL	5
SOAL 1	6
A. Source Code.....	7
B. Output Program	17
C. Pembahasan	21
D. Tautan Git	26

DAFTAR GAMBAR

Gambar 1. Tampilan Awal Aplikasi	6
Gambar 2. Tampilan Aplikasi Setelah Dijalankan.....	6
Gambar 3. Screenshot Hasil Jawaban Soal 1	17
Gambar 4. Screenshot Hasil Jawaban Soal 1	17
Gambar 5. Screenshot Hasil Jawaban Soal 1	18
Gambar 6. Screenshot Hasil Jawaban Soal 1	18
Gambar 7. Screenshot Hasil Jawaban Soal 1	19
Gambar 8. Screenshot Hasil Jawaban Soal 1	19
Gambar 9. Screenshot Hasil Jawaban Soal 1	20
Gambar 10. Screenshot Hasil Jawaban Soal 1	20
Gambar 11. Screenshot Hasil Jawaban Soal 1.....	20

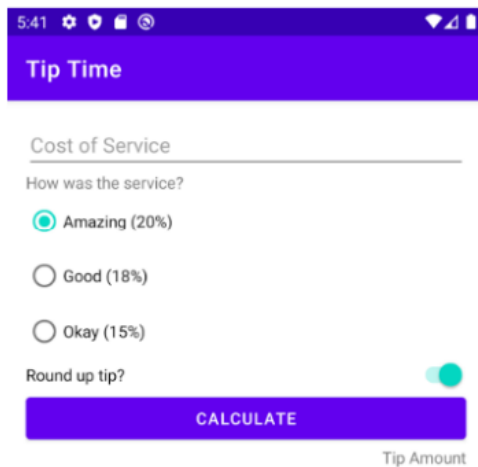
DAFTAR TABEL

Table 1. Source Code Jawaban Soal 1	8
Table 2. Source Code Jawaban Soal 1	13
Table 3. Source Code Jawaban Soal 1	15
Table 4. Source Code Jawaban Soal 1	16
Table 5. Source Code Jawaban Soal 1	16
Table 6. Source Code Jawaban Soal 1	16

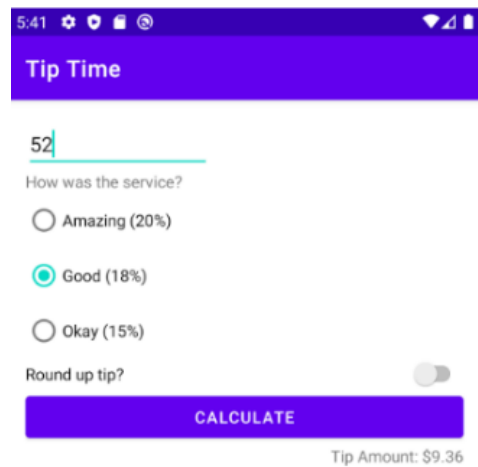
SOAL 1

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

1. Input Biaya Layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
2. Pilihan Persentase Tip: Pengguna dapat memilih persentase tip yang diinginkan dari opsi yang disediakan, yaitu 15%, 18%, dan 20%.
3. Pengaturan Pembulatan Tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
4. Tampilan Hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.:



Gambar 1. Tampilan Awal Aplikasi



Gambar 2. Tampilan Aplikasi Setelah Dijalankan

A. Source Code

File MainActivity.kt	
1	package com.example.tip_time
2	
3	import android.os.Bundle
4	import android.widget.Toast
5	import androidx.appcompat.app.AppCompatActivity
6	import androidx.core.view.WindowCompat
7	import com.example.tip_time.databinding.ActivityMainBinding
8	import java.text.NumberFormat
9	import kotlin.math.ceil
10	
11	class MainActivity : AppCompatActivity() {
12	
13	private lateinit var binding: ActivityMainBinding
14	
15	override fun onCreate(savedInstanceState: Bundle?) {
16	super.onCreate(savedInstanceState)
17	
18	WindowCompat.setDecorFitsSystemWindows(window,
19	false)
20	binding =
21	ActivityMainBinding.inflate(layoutInflater)
22	setContentView(binding.root)
23	binding.btnCalculate.setOnClickListener {
24	calculateTip() }
25	}
26	private fun calculateTip() {

27	val stringInTextField =
	binding.etCostOfService.text.toString()
28	val cost = stringInTextField.toDoubleOrNull()
29	if (cost == null cost == 0.0) {
30	binding.etCostOfService.error = "Please enter
	a valid amount"
	Toast.makeText(this, "Invalid input. Please
31	enter a number.", Toast.LENGTH_SHORT).show()
32	binding.tvTipAmount.text = ""
33	return
34	}
35	
36	val tipPercentage = when
	(binding.rgTipOptions.checkedRadioButtonId) {
37	R.id.rb_amazing -> 0.20
38	R.id.rb_good -> 0.18
39	else -> 0.15
40	}
41	
42	var tip = tipPercentage * cost
43	if (binding.switchRoundUp.isChecked) {
44	tip = ceil(tip)
45	}
46	
47	val formattedTip =
	NumberFormat.getCurrencyInstance().format(tip)
48	binding.tvTipAmount.text =
	getString(R.string.tip_amount, formattedTip)
49	}
50	}

Table 1. Source Code Jawaban Soal 1

File activity_main.xml	
1	<?xml version="1.0" encoding="utf-8"?>
2	<ScrollView
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"
4	android:layout_width="match_parent"
5	android:layout_height="match_parent"
6	android:background="#EBE6F3"
7	android:paddingTop="?attr/actionBarSize"
8	android:padding="16dp">
9	
10	<androidx.constraintlayout.widget.ConstraintLayout
11	android:layout_width="match_parent"
12	android:layout_height="wrap_content">
13	
14	<TextView
15	android:id="@+id/titleText"
16	android:layout_width="match_parent"
17	android:layout_height="wrap_content"
18	android:background="#8000FF"
19	android:padding="16dp"
20	android:text="Tip Time"
21	android:textColor="#FFFFFF"
22	android:textSize="24sp"
23	android:textStyle="bold"
24	app:layout_constraintEnd_toEndOf="parent"
25	app:layout_constraintStart_toStartOf="parent"
	"
26	app:layout_constraintTop_toTopOf="parent" />
27	

28	<EditText
29	android:id="@+id/et_cost_of_service"
30	android:layout_width="0dp"
31	android:layout_height="wrap_content"
32	android:hint="Cost of Service"
33	android:inputType="numberDecimal"
34	android:textSize="18sp"
35	android:layout_marginTop="24dp"
36	android:background="@android:drawable/edit_t
	ext"
37	android:padding="8dp"
38	app:layout_constraintTop_toBottomOf="@id/tit
	leText"
39	app:layout_constraintStart_toStartOf="parent
	"
40	app:layout_constraintEnd_toEndOf="parent" />
41	
42	<TextView
43	android:id="@+id/textViewHowWasService"
44	android:layout_width="wrap_content"
45	android:layout_height="wrap_content"
46	android:text="How was the service?"
47	android:textSize="16sp"
48	android:textColor="#999999"
49	android:layout_marginTop="16dp"
50	app:layout_constraintTop_toBottomOf="@id/et_
	cost_of_service"
51	app:layout_constraintStart_toStartOf="parent
	" />
52	
53	<RadioGroup

54	android:id="@+id/rg_tip_options"
55	android:layout_width="0dp"
56	android:layout_height="wrap_content"
57	android:orientation="vertical"
58	app:layout_constraintTop_toBottomOf="@id/textViewHowWasService"
59	app:layout_constraintStart_toStartOf="parent"
60	"
61	app:layout_constraintEnd_toEndOf="parent">
62	<RadioButton
63	android:id="@+id/rb_amazing"
64	android:layout_width="wrap_content"
65	android:layout_height="wrap_content"
66	android:text="Amazing (20%) "
67	android:textColor="@color/black"/>
68	
69	<RadioButton
70	android:id="@+id/rb_good"
71	android:layout_width="wrap_content"
72	android:layout_height="wrap_content"
73	android:text="Good (18%) "
74	android:textColor="@color/black"/>
75	
76	<RadioButton
77	android:id="@+id/rb_okay"
78	android:layout_width="wrap_content"
79	android:layout_height="wrap_content"
80	android:text="Okay (15%) "
81	android:textColor="@color/black"/>
82	</RadioGroup>

83	
84	<pre> <TextView </pre>
85	<pre> android:id="@+id/textViewRoundUp" </pre>
86	<pre> android:layout_width="0dp" </pre>
87	<pre> android:layout_height="wrap_content" </pre>
88	<pre> android:text="Round up tip?" </pre>
89	<pre> android:textSize="16sp" </pre>
90	<pre> app:layout_constraintTop_toBottomOf="@id/rg_ tip_options" </pre>
91	<pre> app:layout_constraintStart_toStartOf="parent " </pre>
92	<pre> app:layout_constraintEnd_toStartOf="@id/swit ch_round_up" </pre>
93	<pre> app:layout_constraintHorizontal_bias="0" </pre>
94	<pre> android:layout_marginTop="8dp" /> </pre>
95	
96	<pre> <Switch </pre>
97	<pre> android:id="@+id/switch_round_up" </pre>
98	<pre> android:layout_width="wrap_content" </pre>
99	<pre> android:layout_height="wrap_content" </pre>
100	<pre> android:checked="false" </pre>
101	<pre> android:thumbTint="@drawable/switch_thumb_co lor" </pre>
102	<pre> android:trackTint="@drawable/switch_track_co lor" </pre>
103	<pre> app:layout_constraintTop_toTopOf="@id/textVi ewRoundUp" </pre>
104	<pre> app:layout_constraintBottom_toBottomOf="@id/ textViewRoundUp" </pre>
105	<pre> app:layout_constraintEnd_toEndOf="parent" /> </pre>
106	

107	<Button
108	android:id="@+id/btn_calculate"
109	android:layout_width="0dp"
110	android:layout_height="wrap_content"
111	android:text="CALCULATE"
112	android:textColor="#FFFFFF"
113	android:layout_marginTop="16dp"
114	app:layout_constraintTop_toBottomOf="@id/textViewRoundUp"
115	app:layout_constraintStart_toStartOf="parent"
116	app:layout_constraintEnd_toEndOf="parent"/>
117	
118	<TextView
119	android:id="@+id/tv_tip_amount"
120	android:layout_width="wrap_content"
121	android:layout_height="wrap_content"
122	android:text="Tip Amount: \$0.00"
123	android:textSize="16sp"
124	android:textColor="#888888"
125	android:layout_marginTop="16dp"
126	app:layout_constraintTop_toBottomOf="@id/btn_calculate"
127	app:layout_constraintEnd_toEndOf="parent" />
128	
129	</androidx.constraintlayout.widget.ConstraintLayout>
130	</ScrollView>

Table 2. Source Code Jawaban Soal 1

File build.gradle.kts (:app)

```
1  plugins {
2      alias(libs.plugins.android.application)
3      alias(libs.plugins.kotlin.android)
4  }
5
6  android {
7      namespace = "com.example.tip_time"
8      compileSdk = 35
9
10     defaultConfig {
11         applicationId = "com.example.dice_roll"
12         minSdk = 24
13         targetSdk = 35
14         versionCode = 1
15         versionName = "1.0"
16
17         testInstrumentationRunner =
18             "androidx.test.runner.AndroidJUnitRunner"
19     }
20     buildFeatures{
21         viewBinding = true
22     }
23
24     buildTypes {
25         release {
26             isMinifyEnabled = false
27             proguardFiles(
28                 getDefaultProguardFile("proguard-android-
optimize.txt"),
```

29	"proguard-rules.pro"
30)
31	}
32	}
33	compileOptions {
34	sourceCompatibility = JavaVersion.VERSION_11
35	targetCompatibility = JavaVersion.VERSION_11
36	}
37	kotlinOptions {
38	jvmTarget = "11"
39	}
40	}
41	
42	dependencies {
43	implementation(libs.androidx.core.ktx)
44	implementation(libs.androidx.appcompat)
45	implementation(libs.material)
46	implementation(libs.androidx.activity)
47	implementation(libs.androidx.constraintlayout)
48	testImplementation(libs.junit)
49	androidTestImplementation(libs.androidx.junit)
50	androidTestImplementation(libs.androidx.espresso.core)
51	}

Table 3. Source Code Jawaban Soal 1

File string.xml	
1	<resources>
2	<string name="app_name">Tip_Time</string>
3	<string name="cost_of_service">Cost of Service</string>
4	<string name="how_was_service">How was the

	service?</string>
5	<string name="amazing">Amazing (20%)</string>
6	<string name="good">Good (18%)</string>
7	<string name="okay">Okay (15%)</string>
8	<string name="round_up_tip">Round up tip?</string>
9	<string name="calculate">CALCULATE</string>
10	<string name="tip_amount">Tip Amount: %s</string>
11	<string name="default_tip_amount">Tip Amount:
	\$0.00</string>
12	</resources>

Table 4. Source Code Jawaban Soal 1

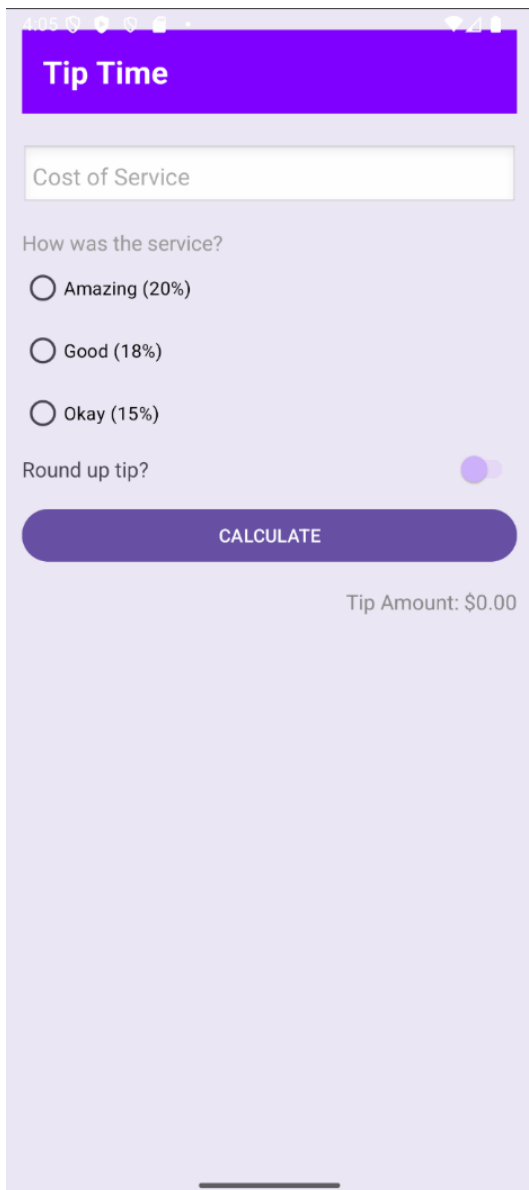
File switch_track_color.xml	
1	<?xml version="1.0" encoding="utf-8"?>
2	<selector
	xmlns:android="http://schemas.android.com/apk/res/android">
3	<item android:color="#8000FF"
	android:state_checked="true"/>
4	<item android:color="#D1B3FF"/>
5	</selector>

Table 5. Source Code Jawaban Soal 1

File switch_thumb_color.xml	
1	<?xml version="1.0" encoding="utf-8"?>
2	<selector
	xmlns:android="http://schemas.android.com/apk/res/android">
3	<item android:color="#62339C"
	android:state_checked="true"/>
4	<item android:color="#D1B3FF"/>
5	</selector>

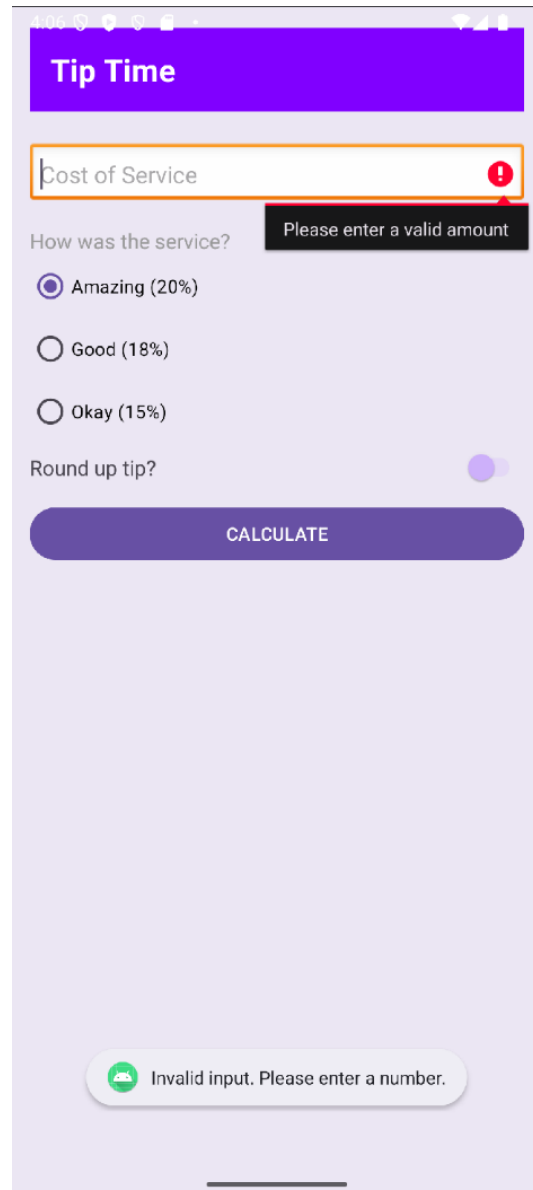
Table 6. Source Code Jawaban Soal 1

B. Output Program



The screenshot shows the 'Tip Time' app interface. At the top is a purple header with the text 'Tip Time'. Below it is a white input field labeled 'Cost of Service'. Underneath is the question 'How was the service?' followed by three radio button options: 'Amazing (20%)', 'Good (18%)', and 'Okay (15%)'. To the right of these options is a toggle switch for 'Round up tip?'. Below the radio buttons is a purple 'CALCULATE' button. At the bottom right, the text 'Tip Amount: \$0.00' is displayed.

Gambar 3. Screenshot Hasil Jawaban Soal 1



This screenshot shows the same 'Tip Time' app interface as Gambar 3, but with an error state. The 'Cost of Service' input field is highlighted with a red border and a red exclamation mark icon. A black tooltip with the text 'Please enter a valid amount' is visible next to the input field. The 'Amazing (20%)' radio button is now selected. At the bottom, a green toast message with a speech bubble icon says 'Invalid input. Please enter a number.'.

Gambar 4. Screenshot Hasil Jawaban Soal 1

4:06

Tip Time

52

How was the service?

☒ Amazing (20%)

☐ Good (18%)

☐ Okay (15%)

Round up tip? ☐

CALCULATE

Tip Amount: \$10.40

Gambar 5. Screenshot Hasil Jawaban Soal 1

5:29

Tip Time

52

How was the service?

☒ Amazing (20%)

☐ Good (18%)

☐ Okay (15%)

Round up tip? ☒

CALCULATE

Tip Amount: \$11.00

Gambar 6. Screenshot Hasil Jawaban Soal 1

4:07

Tip Time

52

How was the service?

☐ Amazing (20%)

☒ Good (18%)

☐ Okay (15%)

Round up tip? ☐

CALCULATE

Tip Amount: \$9.36

Gambar 7. Screenshot Hasil Jawaban Soal 1

5:30

Tip Time

52

How was the service?

☐ Amazing (20%)

☒ Good (18%)

☐ Okay (15%)

Round up tip? ☒

CALCULATE

Tip Amount: \$10.00

Gambar 8. Screenshot Hasil Jawaban Soal 1

5:29

Tip Time

52

How was the service?

☐ Amazing (20%)

☐ Good (18%)

☒ Okay (15%)

Round up tip? ☐

CALCULATE

Tip Amount: \$7.80

Gambar 9. Screenshot Hasil Jawaban Soal 1

5:31

Tip Time

52

How was the service?

☐ Amazing (20%)

☐ Good (18%)

☒ Okay (15%)

Round up tip? ☒

CALCULATE

Tip Amount: \$8.00

Gambar 10. Screenshot Hasil Jawaban Soal 1

3:52

Tip Time

52

How was the service?

☒ Amazing (20%)

☐ Good (18%)

☐ Okay (15%)

Round up tip? ☒

CALCULATE

Gambar 11. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

File MainActivity.kt

Kelas utama **MainActivity** mewarisi **AppCompatActivity**, yang merupakan kelas dasar untuk aktivitas yang mendukung fitur-fitur dari Android. Di dalam metode **onCreate()**, aktivitas menginisialisasi tampilan dengan memanfaatkan **ActivityMainBinding**, yaitu bagian dari mekanisme **View Binding**. Mekanisme ini memungkinkan kita untuk berinteraksi dengan elemen UI secara langsung tanpa perlu memanggil **findViewById**, sehingga membuat kode lebih aman dari kesalahan tipe dan lebih mudah dibaca.

Pemanggilan **WindowCompat.setDecorFitsSystemWindows(window, false)** berfungsi untuk mengatur apakah tata letak aktivitas akan menyesuaikan dengan batas sistem. Jika mengatur ke **false**, konten aplikasi dapat meluas ke seluruh layar, yang biasanya digunakan untuk tampilan imersif atau mendukung desain penuh layar.

Setelah tampilan aktivitas diatur, sebuah **OnClickListener** dipasang ke tombol **btnCalculate** melalui **binding.btnCalculate.setOnClickListener { calculateTip() }**. Ini berarti ketika tombol ditekan, fungsi **calculateTip()** akan dijalankan. Fungsi inilah yang berisi logika utama dari aplikasi penghitung tip.

Di dalam **calculateTip()**, teks dari **EditText (etCostOfService)** diambil dan diubah menjadi **Double** menggunakan **toDoubleOrNull()**, yang akan menghasilkan **null** jika konversi gagal. Jika nilai input tidak valid, maka dua hal seperti **EditText** akan menampilkan error menggunakan **setError()** dengan pesan "**Please enter a valid amount**", dan sebuah **Toast** akan muncul, memberi tahu pengguna bahwa input tidak valid.

Selain itu, jika input tidak valid, tampilan **tvTipAmount** dikosongkan agar tidak menampilkan hasil yang menyesatkan dari input sebelumnya. Jika input valid, aplikasi menentukan persentase tip berdasarkan tombol radio yang dipilih yaitu, **R.id.rb_amazing** untuk **20%**, **R.id.rb_good** untuk **18%**, dan **R.id.rb_okay** untuk **15%**.

Setelah persentase tip ditentukan, aplikasi mengalikan nilai **cost** dengan persentase tip untuk mendapatkan jumlah tip. Jika pengguna mencentang atau menyalakan

switchRoundUp, maka nilai tip dibulatkan ke atas menggunakan fungsi **ceil()** dari Kotlin yang digunakan untuk membulatkan angka desimal ke bilangan bulat terdekat ke atas.

Terakhir, hasil tip diformat ke dalam bentuk mata uang lokal menggunakan **NumberFormat.getCurrencyInstance().format(tip)**. Hasil ini kemudian ditampilkan pada **TextView(tvTipAmount)** menggunakan **getString()** dengan referensi ke string di **strings.xml** yang memiliki format seperti **"Tip amount: %s"**

File activity_main.xml

Layout aplikasi dibungkus oleh elemen **ScrollView** agar tampilan bisa digulir secara vertikal ketika konten melebihi tinggi layar. **ScrollView** ini memiliki lebar dan tinggi yang disesuaikan dengan ukuran layar **match_parent**, dengan latar belakang berwarna **#EBE6F3**. Padding atas ditentukan berdasarkan tinggi **ActionBar**, sedangkan padding umum diatur sebesar **16dp** untuk menjaga jarak antara konten dan tepi layar. Di dalam **ScrollView**, digunakan **ConstraintLayout** sebagai kontainer utama yang fleksibel dalam mengatur posisi elemen UI berdasarkan constraint antar elemen.

Komponen pertama adalah **TextView** dengan ID **titleText** yang berfungsi sebagai judul aplikasi. Komponen ini menampilkan teks **"Tip Time"** dengan gaya tebal, ukuran font **24sp**, latar belakang **#8000FF**, dan teks berwarna putih. Elemen ini direntangkan dari kiri ke kanan dan diletakkan di bagian atas layout. Di bawahnya terdapat **EditText** dengan ID **et_cost_of_service** sebagai tempat pengguna menginput biaya layanan. **EditText** ini memiliki *hint* **"Cost of Service"**, menerima input angka desimal, dan menggunakan latar belakang bawaan Android dengan padding **8dp**. Komponen ini juga diatur agar lebarnya mengikuti constraint kiri dan kanan serta memiliki margin atas sebesar **24dp**.

Selanjutnya, terdapat **TextView** dengan ID **textViewHowWasService** yang memberikan petunjuk kepada pengguna dengan teks **"How was the service?"** berwarna abu-abu terang. Di bawah petunjuk ini terdapat **RadioGroup** dengan ID **rg_tip_options** yang berisi tiga **RadioButton**, yaitu **rb_amazing**, **rb_good**,

dan **rb_okay**, yang masing-masing menampilkan label “**Amazing (20%)**”, “**Good (18%)**”, dan “**Okay (15%)**” dengan warna teks hitam. Komponen ini digunakan untuk memilih tingkat kepuasan pengguna terhadap layanan.

Setelah itu, terdapat pasangan **TextView** dan **Switch** yang digunakan untuk opsi pembulatan tip. **TextView** dengan ID **textViewRoundUp** menampilkan pertanyaan “**Round up tip?**”, sementara **Switch** dengan ID **switch_round_up** memungkinkan pengguna mengaktifkan atau menonaktifkan pembulatan ke atas. **Switch** ini menggunakan warna **thumb** dan **track** yang ditentukan oleh file **drawable** khusus **switch_thumb_color** dan **switch_track_color**. Kedua komponen ini dirancang agar tampil sejajar secara horizontal.

Berikutnya adalah **Button** dengan ID **btn_calculate** yang berfungsi untuk memicu perhitungan tip ketika ditekan. Tombol ini memiliki teks “**CALCULATE**” dengan warna putih, margin atas **16dp**, dan direntangkan secara horizontal di bawah elemen **switch**. Terakhir, **TextView** dengan ID **tv_tip_amount** digunakan untuk menampilkan hasil perhitungan tip dengan teks default “**Tip Amount: \$0.00**”, teks berwarna abu-abu, dan diletakkan di bawah tombol kalkulasi.

File build.gradle.kts (:app)

Pada file **build.gradle.kts (:app)** yang ditampilkan adalah berkas konfigurasi utama untuk proyek Android berbasis Kotlin menggunakan Gradle dengan DSL (*Domain Specific Language*) Kotlin. File ini digunakan untuk mendefinisikan bagaimana proyek akan dibangun, termasuk plugin yang digunakan, versi SDK Android, fitur *build*, dan dependensi *library* yang dibutuhkan dalam pengembangan aplikasi.

Bagian pertama dari file ini adalah deklarasi plugin, di mana proyek ini menggunakan dua plugin utama: **android.application** dan **kotlin.android**. Plugin **android.application** digunakan untuk menandai bahwa proyek ini adalah sebuah aplikasi Android, bukan *library*. Sementara itu, plugin **kotlin.android** menambahkan dukungan Kotlin ke dalam proyek Android, sehingga pengembangan dapat dilakukan menggunakan bahasa Kotlin. Penulisan **alias(libs.plugins...)** menunjukkan bahwa proyek ini menggunakan fitur *Version Catalog*, yang memungkinkan

pengelolaan versi plugin dan dependensi secara terpusat melalui file **libs.versions.toml**.

Selanjutnya, blok **android** berisi konfigurasi utama proyek. Properti **namespace** menentukan *namespace* atau ruang nama dasar yang akan digunakan oleh aplikasi, umumnya mengikuti format nama paket. Kemudian **compileSdk** diatur ke **versi 35**, yang berarti aplikasi ini dikompilasi menggunakan **Android API level 35**, memberikan akses terhadap fitur-fitur terbaru dari Android versi tersebut. Di dalam **defaultConfig**, terdapat beberapa pengaturan penting seperti **applicationId** yang menjadi identitas unik aplikasi di perangkat dan di *Google Play Store*. **minSdk** diatur ke 24, artinya aplikasi ini hanya dapat diinstal di perangkat dengan **Android Nougat (7.0) ke atas**. **targetSdk** ditetapkan ke 35 agar aplikasi ini dioptimalkan untuk versi Android terbaru. Selain itu, **versionCode** dan **versionName** digunakan untuk pengelolaan versi aplikasi, di mana **versionCode** bersifat **numerik** dan **dibutuhkan saat proses update di Google Play**, sedangkan **versionName** bersifat **deskriptif** dan **ditampilkan kepada pengguna**. **testInstrumentationRunner** menentukan runner untuk pengujian instrumentasi Android.

Pada bagian **buildFeatures**, fitur **viewBinding** diaktifkan. **ViewBinding** adalah fitur yang memungkinkan pengembang untuk lebih mudah mengakses elemen antarmuka pengguna (UI) tanpa harus menggunakan **findViewById**, sehingga lebih aman dari kesalahan *runtime* dan meningkatkan produktivitas. Kemudian di blok **buildTypes**, terdapat konfigurasi untuk tipe build **release**, yang biasanya digunakan saat aplikasi akan dipublikasikan. Di sini, **isMinifyEnabled** diatur ke **false**, artinya proses minifikasi kode tidak diaktifkan, yang umum dilakukan pada tahap awal pengembangan atau untuk aplikasi kecil. File **proguardFiles** menunjuk ke konfigurasi *default ProGuard* dan file tambahan **proguard-rules.pro** untuk aturan khusus.

Bagian **compileOptions** dan **kotlinOptions** memastikan bahwa proyek menggunakan Java 11 untuk kompatibilitas sumber dan target *bytecode*, dan juga mengarahkan Kotlin untuk menggunakan JVM target versi 11.

Terakhir, bagian **dependencies** mencantumkan semua *library* yang digunakan dalam aplikasi. Di antaranya adalah **androidx.core.ktx**, **appcompat**, **material**,

activity, dan **constraintlayout**, yang merupakan *library* penting untuk pengembangan antarmuka dan fungsi dasar aplikasi Android. Untuk pengujian, digunakan **junit** untuk unit *testing* standar, serta **androidx.junit** dan **espresso.core** untuk *instrumented testing* yang berjalan langsung di perangkat atau emulator. Semua dependensi ini juga ditulis menggunakan **libs.xxx**, menunjukkan bahwa proyek mengadopsi pendekatan *Version Catalog* yang lebih modern dan terstruktur dalam pengelolaan dependensi.

File String.xml

Setiap elemen dalam file ini didefinisikan menggunakan tag **<string>** dengan atribut **name** yang unik, yang nantinya akan digunakan sebagai referensi dari file layout XML atau file Kotlin/Java. **<string name="app_name">Tip_Time</string>** menetapkan nama aplikasi sebagai "**Tip_Time**".

String seperti "**cost_of_service**", "**how_was_service**", "**amazing**", "**good**", dan "**okay**" merupakan label teks yang digunakan dalam tampilan antarmuka pengguna (UI), yang masing-masing ditampilkan sebagai petunjuk input atau label untuk tombol radio.

Selanjutnya, string "**round_up_tip**" dan "**calculate**" masing-masing digunakan sebagai label untuk opsi *switch* dan tombol kalkulasi. String "**tip_amount**" menggunakan format string **%s** yang akan digantikan oleh nilai dinamis saat aplikasi berjalan, yaitu jumlah tip yang telah diformat sebagai mata uang. Ini memungkinkan penggunaan satu string *template* yang fleksibel dan dapat disesuaikan selama *runtime*. Adapun string "**default_tip_amount**" berisi teks default yang ditampilkan saat belum ada perhitungan dilakukan, yaitu "**Tip Amount: \$0.00**".

File switch_track_color.xml

Elemen **<selector>**, memungkinkan pengembang untuk mendefinisikan warna yang berbeda tergantung pada kondisi tertentu dari elemen tersebut, misalnya saat elemen sedang dipilih (*checked*), difokuskan, ditekan, atau dalam keadaan normal.

Terdapat dua elemen `<item>`. Elemen pertama memiliki atribut `android:state_checked="true"`, artinya kondisi ini berlaku ketika elemen UI berada dalam keadaan terpilih (*checked*). Jika kondisi ini terpenuhi, maka warna yang digunakan adalah `#8000FF`, yaitu warna ungu tua. Warna ini akan terlihat, misalnya, pada teks atau ikon `RadioButton` yang dipilih, tergantung pada di mana *color selector* ini diterapkan.

Elemen kedua bertindak sebagai *default* atau *fallback*. Artinya, ketika elemen tidak dalam keadaan *checked*, maka warna yang digunakan adalah `#D1B3FF`, yaitu ungu muda. Ini memberikan efek visual bahwa elemen yang tidak aktif tampil lebih lembut, sedangkan elemen aktif tampil lebih mencolok.

File switch_thumb_color.xml

Elemen `<selector>`, memungkinkan pengembang untuk mendefinisikan warna yang berbeda tergantung pada kondisi tertentu dari elemen tersebut, misalnya saat elemen sedang dipilih (*checked*), difokuskan, ditekan, atau dalam keadaan normal.

Terdapat dua elemen `<item>`. Elemen pertama memiliki atribut `android:state_checked="true"`, artinya kondisi ini berlaku ketika elemen UI berada dalam keadaan terpilih (*checked*). Jika kondisi ini terpenuhi, maka warna yang digunakan adalah `#62339C`, yaitu warna ungu gelap.

Elemen kedua tidak memiliki atribut `state`, bertindak sebagai *default*. Artinya, ketika elemen tidak dalam keadaan *checked*, maka warna yang digunakan adalah `#D1B3FF`, yaitu ungu muda. Sehingga pengguna dapat dengan mudah membedakan antara elemen yang aktif dan tidak aktif secara visual.

D. Tautan Git

<https://github.com/biilaaa/Pemrograman-Mobile>