LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 2

ANDROID LAYOUT

Oleh:

Muhammad Fauzan Ahsani **NIM. 2310817310009**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
MODUL 2

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

Nama Praktikan : Muhammad Fauzan Ahsani
NIM : 2310817310009

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

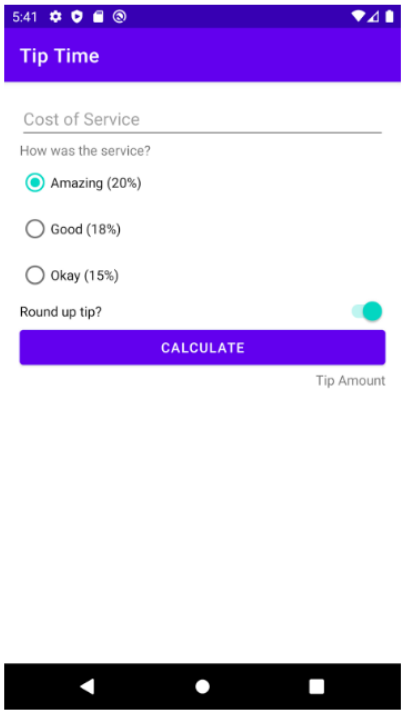
DAFTAR GAMBAR

DAFTAR TABEL

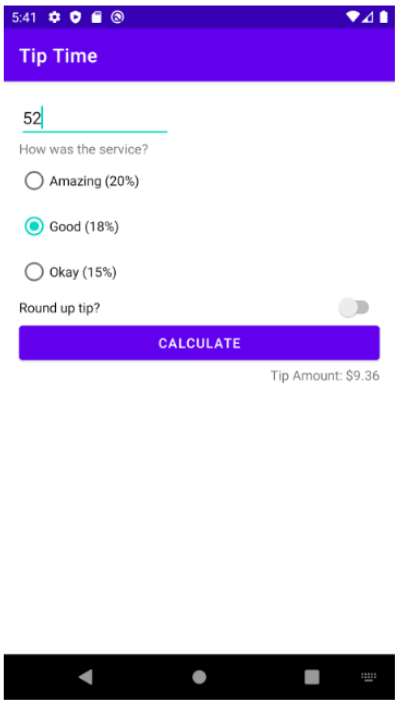
SOAL 1

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

1. MainActivity.kt

Tabel 1. Source Code MainActivity.kt

```
 1 package com.example.tipcalculator
 2
 3 import android.os.Bundle
 4 import androidx.activity.ComponentActivity
 5 import androidx.activity.compose.setContent
 6 import androidx.compose.runtime.Composable
 7 import androidx.compose.ui.tooling.preview.Preview
 8 import androidx.lifecycle.viewmodel.compose.viewModel
 9 import com.example.tipcalculator.ui.TipCalculatorScreen
10 import
   com.example.tipcalculator.viewmodel.TipCalculatorViewMo
   del
11 import
   com.example.tipcalculator.ui.theme.TipCalculatorTheme
12
13 class MainActivity : ComponentActivity() {
14     override fun onCreate(savedInstanceState: Bundle?)
   {
```

```
15          super.onCreate(savedInstanceState)
16          setContent {
17              TipCalculatorTheme {
18                  val tipViewModel:
    TipCalculatorViewModel = viewModel()
19                  TipCalculatorScreen(viewModel =
    tipViewModel)
20              }
21          }
22      }
23  }
24
25  @Preview(showBackground = true, showSystemUi = true)
26  @Composable
27  fun TipCalculatorPreview() {
28      TipCalculatorTheme {
29          val previewViewModel: TipCalculatorViewModel =
    viewModel()
30          TipCalculatorScreen(viewModel =
    previewViewModel)
31      }
32  }
```

2. TipCalculatorViewModel.kt

Tabel 2. Source Code TipCalculatorViewModel.kt

```
 1  package com.example.tipcalculator.viewmodel
 2
 3  import androidx.lifecycle.ViewModel
 4  import kotlinx.coroutines.flow.MutableStateFlow
 5  import kotlinx.coroutines.flow.StateFlow
 6  import kotlin.math.ceil
 7
 8  class TipCalculatorViewModel: ViewModel() {
 9      private val _billAmt = MutableStateFlow("")
10      val billAmt: StateFlow<String> = _billAmt
11
12      private val _rating = MutableStateFlow("Okay")
13      val rating: StateFlow<String> = _rating
14
15      private val _tipAmt = MutableStateFlow(0f)
16      val tipAmt: StateFlow<Float> = _tipAmt
17
18      private val _roundUp = MutableStateFlow(false)
19      val roundUp: StateFlow<Boolean> = _roundUp
20
```

```
21      fun onBillChanged(newAmt: String){
22          _billAmt.value = newAmt
23      }
24
25      fun onRatingSelected(newRating: String){
26          _rating.value = newRating
27      }
28
29      fun onRoundUpChanged(newValue: Boolean) {
30          _roundUp.value = newValue
31      }
32
33      fun calculateTip() {
34          val bill = _billAmt.value.replace(",",
    "").toDoubleOrNull() ?: 0.0
35          val percentage = when (_rating.value) {
36              "Okay (15%)" -> 15.0
37              "Good (18%)" -> 18.0
38              "Amazing (20%)" -> 20.0
39              else -> 0.0
40          }
41          val tip = bill * percentage / 100.0
42          _tipAmt.value = if (_roundUp.value)
    ceil(tip).toFloat() else tip.toFloat()
43      }
44  }
```

3. TipCalculatorScreen.kt

Tabel 3. Source Code TipCalculatorScreen.kt

```
 1  package com.example.tipcalculator.ui
 2
 3  import androidx.compose.foundation.layout.*
 4  import androidx.compose.material3.*
 5  import androidx.compose.runtime.*
 6  import androidx.compose.ui.Modifier
 7  import androidx.compose.ui.tooling.preview.Preview
 8  import androidx.compose.ui.unit.dp
 9  import androidx.lifecycle.viewmodel.compose.viewModel
10  import com.example.tipcalculator.ui.components.*
11  import
    com.example.tipcalculator.viewmodel.TipCalculatorViewM
    odel
12  import androidx.compose.ui.text.font.*
13  import java.text.DecimalFormat
14  import androidx.compose.foundation.rememberScrollState
```

```
15 import androidx.compose.foundation.verticalScroll
16
17
18 @OptIn(ExperimentalMaterial3Api::class)
19 @Composable
   fun TipCalculatorScreen(viewModel:
20 TipCalculatorViewModel = viewModel()) {
       val bill by viewModel.billAmt.collectAsState()
21     val rating by viewModel.rating.collectAsState()
22     val tip by viewModel.tipAmt.collectAsState()
23     val roundUp by viewModel.roundUp.collectAsState()
24
25     val ratingOptions = listOf("Okay (15%)", "Good
26 (18%)", "Amazing (20%)")

27     Scaffold(
28         topBar = {
29             TopAppBar(
30                 colors =
31 TopAppBarDefaults.topAppBarColors(
                       containerColor =
32 MaterialTheme.colorScheme.primaryContainer,
                       titleContentColor =
33 MaterialTheme.colorScheme.primary,
                   ),
34                 title = {
35                     Text("Tip Calculator")
36                 }
37             )
38         },
39         content = { padding ->
40             Column(
41                 modifier = Modifier
42
43 .verticalScroll(rememberScrollState())
                       .padding(padding)
44                     .padding(16.dp)
45                     .fillMaxSize(),
46                 verticalArrangement =
47 Arrangement.spacedBy(16.dp)
             ) {
48                 BillInputField(
49                     value = bill,
50                     onValueChange =
51 viewModel::onBillChanged
                   )
52
```

```
53                  TipRating(
54                      selectedRating = rating,
55                      options = ratingOptions,
56                      onRatingSelected =
57  viewModel::onRatingSelected
                    )
58
59                  Row(
60                      modifier =
61  Modifier.fillMaxWidth(),
                        horizontalArrangement =
62  Arrangement.SpaceBetween
                    ) {
63                      Text("Round Up Tip?")
64                      Switch(
65                          checked = roundUp,
66                          onCheckedChange =
67  viewModel::onRoundUpChanged
                        )
68                  }
69
70                  Button(
71                      onClick = viewModel::calculateTip,
72                      modifier = Modifier.fillMaxWidth()
73                  ) {
74                      Text("Calculate")
75                  }
76                  val formattedTip =
77  formatAsCurrency(tip)

78                  Text(
79                      text = "Tip: \nRp$formattedTip",
80                      style =
81  MaterialTheme.typography.headlineMedium.copy(fontWeigh
    t = FontWeight.Bold)
                    )
82              }
83          }
84      )
85  }
86
87  fun formatAsCurrency(amount: Float): String {
88      val formatter = DecimalFormat("#,###.00")
89      return formatter.format(amount)
90  }
91
92  @Preview(showBackground = true)
```

```
 93 @Composable
 94 fun TipCalculatorScreenPreview() {
 95     val fakeViewModel = TipCalculatorViewModel()
 96     fakeViewModel.onBillChanged("100")
 97     fakeViewModel.onRatingSelected("Amazing")
 98
 99     TipCalculatorScreen(viewModel = fakeViewModel)
100 }
101
```

4. BillInputField.kt

Tabel 4. Source Code BillInputField.kt

```
 1 package com.example.tipcalculator.ui.components
 2
 3 import androidx.compose.material3.*
 4 import androidx.compose.runtime.*
 5 import androidx.compose.ui.Modifier
 6 import androidx.compose.ui.unit.dp
 7 import androidx.compose.foundation.layout.*
 8 import java.text.DecimalFormat
 9 import androidx.compose.foundation.text.KeyboardActions
10 import androidx.compose.foundation.text.KeyboardOptions
11 import androidx.compose.ui.text.TextStyle
12 import androidx.compose.ui.text.font.FontWeight
13 import androidx.compose.ui.text.input.ImeAction
14 import androidx.compose.ui.text.input.KeyboardType
15
16
17 @Composable
18 fun BillInputField(value: String, onValueChange:
   (String) -> Unit) {
19     var formattedValue by remember {
   mutableStateOf(value) }
20
21     OutlinedTextField(
22         modifier = Modifier
23             .fillMaxWidth()
24             .padding(horizontal = 16.dp),
25
26         value = formattedValue,
27         onValueChange = { newValue ->
28             var cleaned =
   newValue.replace(Regex("[^\\d.]"), "")
29             cleaned = cleaned.replace(Regex("\\.+"),
   ".")
```

```
30
31              val parts = cleaned.split(".")
32              val integerPart = parts.getOrElse(0) { "" }
33              val decimalPart = parts.getOrElse(1) { ""
    }.take(2)
34
35              val formattedInteger = try {
36                  when {
37                      integerPart.isEmpty() &&
    decimalPart.isNotEmpty() -> "0"
38                      integerPart.isEmpty() -> ""
39                      else ->
    DecimalFormat("#,###").format(integerPart.toLong())
40                  }
41              } catch (e: NumberFormatException) {
42                  integerPart
43              }
44
45              val formatted = buildString {
46                  append(formattedInteger)
47                  if (decimalPart.isNotEmpty() ||
    cleaned.endsWith(".")) {
48                      append(".$decimalPart")
49                  }
50              }
51
52              formattedValue = formatted
53              onValueChange(formatted)
54          },
55          label = { Text("Enter Bill Amount") },
56          keyboardOptions = KeyboardOptions.Default.copy(
57              keyboardType = KeyboardType.Number,
58              imeAction = ImeAction.Done
59          ),
60          keyboardActions = KeyboardActions.Default,
61          singleLine = true,
62          textStyle = TextStyle.Default.copy(fontWeight =
    FontWeight.Bold)
63      )
64  }
```

5. TipRating.kt

Tabel 5. Source Code TipRating.kt

```
 1 package com.example.tipcalculator.ui.components
 2
 3 import androidx.compose.foundation.layout.*
 4 import androidx.compose.material3.*
 5 import androidx.compose.runtime.*
 6 import androidx.compose.ui.Alignment
 7 import androidx.compose.ui.Modifier
 8 import androidx.compose.ui.unit.dp
 9
10 @Composable
11 fun TipRating(
12     selectedRating: String,
13     options: List<String>,
14     onRatingSelected: (String) -> Unit,
15     modifier: Modifier = Modifier
16 ) {
17     Column(modifier = modifier.fillMaxWidth()) {
18         Text("How was the service?")
19         options.forEach { option ->
20             Row(
21                 verticalAlignment =
   Alignment.CenterVertically
22             ) {
23                 RadioButton(
24                     selected = (option ==
   selectedRating),
25                     onClick = {
   onRatingSelected(option) }
26                 )
27                 Text(option)
28             }
29         }
30     }
31 }
```

6. AndroidManifest.xml

Tabel 6. Source Code AndroidManifest.xml

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2 <manifest
 3 xmlns:android="http://schemas.android.com/apk/res/andro
   id"
 4     xmlns:tools="http://schemas.android.com/tools">
```

```
 5      <application
 6          android:allowBackup="true"
 7
    android:dataExtractionRules="@xml/data_extraction_rules
    "
 8          android:fullBackupContent="@xml/backup_rules"
 9          android:icon="@mipmap/ic_launcher"
10          android:label="@string/app_name"
11          android:roundIcon="@mipmap/ic_launcher_round"
12          android:supportsRtl="true"
13          android:theme="@style/Theme.App.Starting"
14          tools:targetApi="31">
15          <activity
16              android:name=".MainActivity"
17              android:exported="true"
18              android:label="@string/app_name"
19              android:theme="@style/Theme.App.Starting">
20              <intent-filter>
21                  <action
    android:name="android.intent.action.MAIN" />
22
23                  <category
    android:name="android.intent.category.LAUNCHER" />
24              </intent-filter>
25          </activity>
26      </application>
27
28  </manifest>
```

7. themes.xml

Tabel 7. Source Code themes.xml

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <resources>
 3      <style name="Theme.TipCalculator"
    parent="android:Theme.Material.NoActionBar" />
 4  </resources>
```

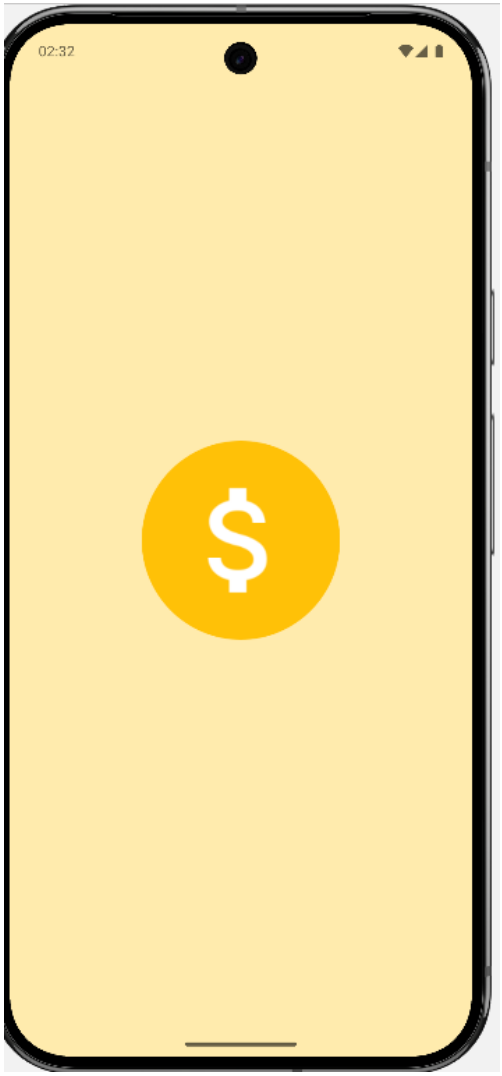
8. styles.xml

Tabel 8. Source Code styles.xml

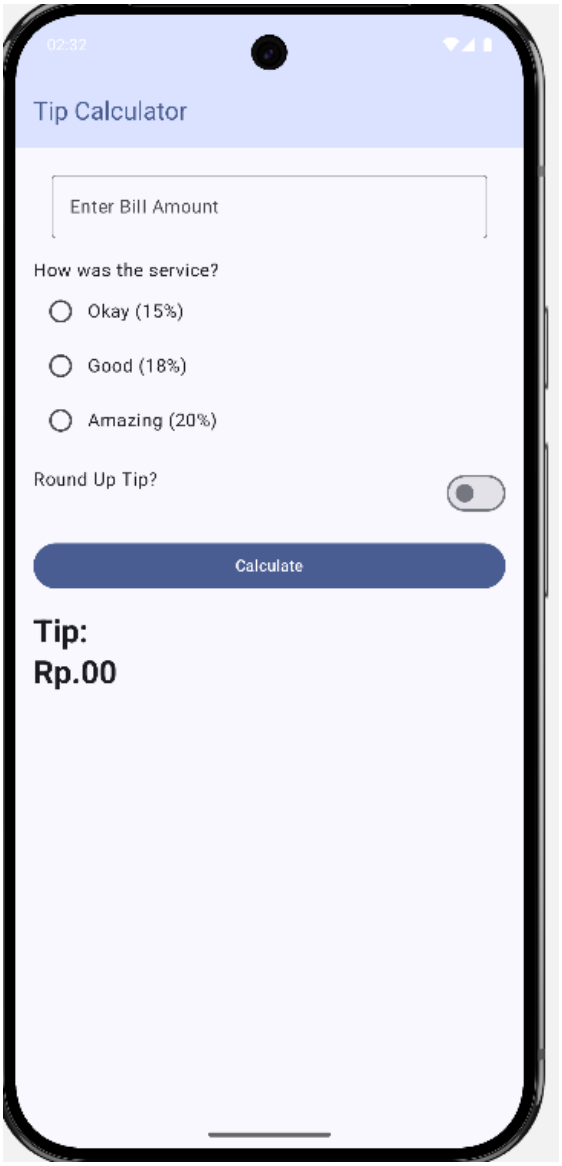
```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <resources>
 3
```

```
    <style name="Theme.App.Starting"
4   parent="Theme.SplashScreen">
        <item
5   name="windowSplashScreenBackground">#ffebad</item>
        <item
    name="windowSplashScreenAnimatedIcon">@mipmap/ic_launch
6   er</item>
        <item
7   name="windowSplashScreenAnimationDuration">200</item>
        <item
    name="postSplashScreenTheme">@style/Theme.TipCalculator
8   </item>
9
10      <item name="android:windowNoTitle">true</item>
11  </style>
    </resources>
```

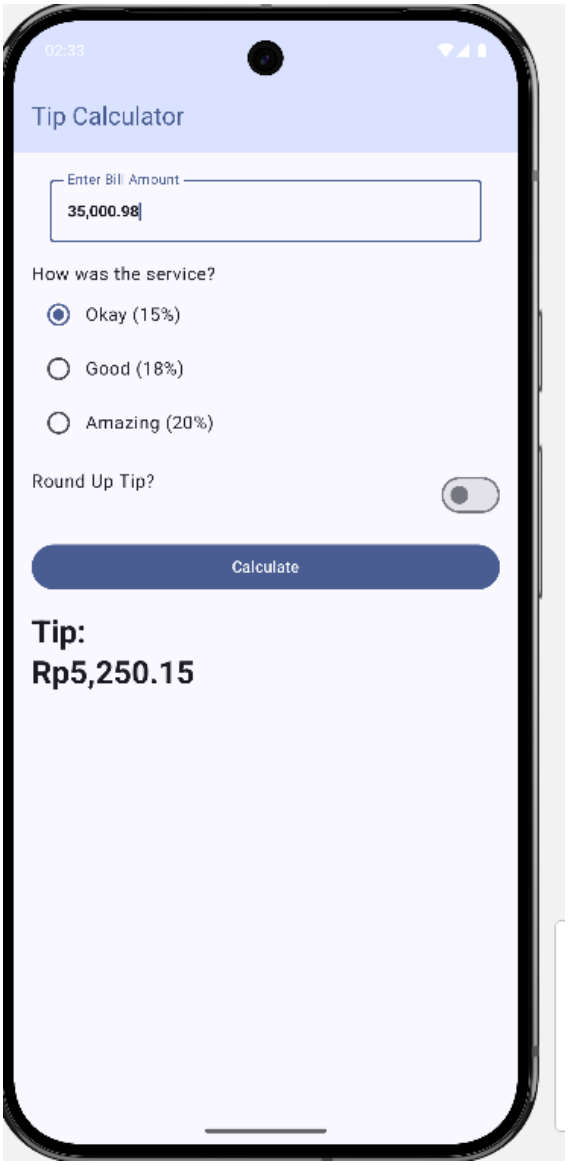
B. Output Program



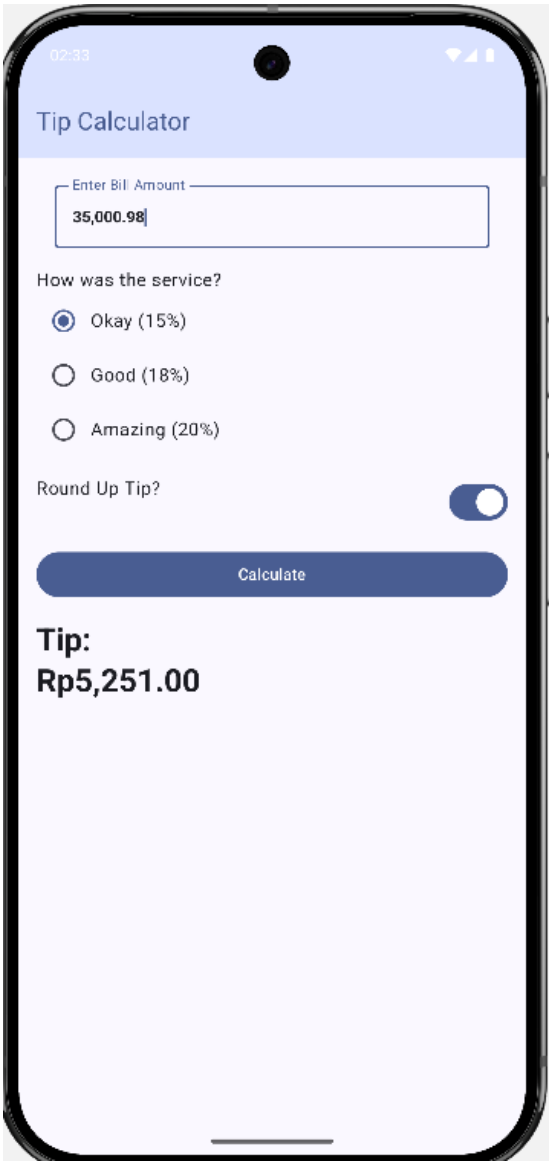
Gambar 3. Splash Screen Aplikasi



Gambar 4. Tampilan Aplikasi Saat Dibuuka Pertama Kali



Gambar 5. Tampilan Aplikasi Saat Semua Field Diisi



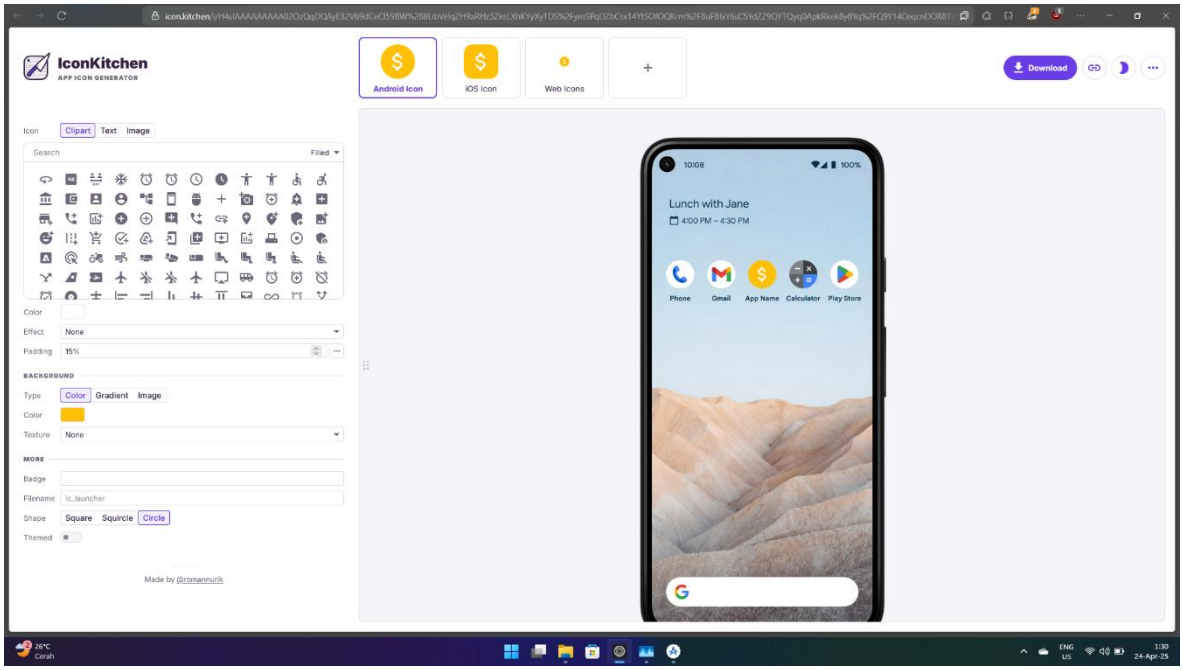
Gambar 6. Tampilan Aplikasi Saat Switch Round Up Diaktifkan

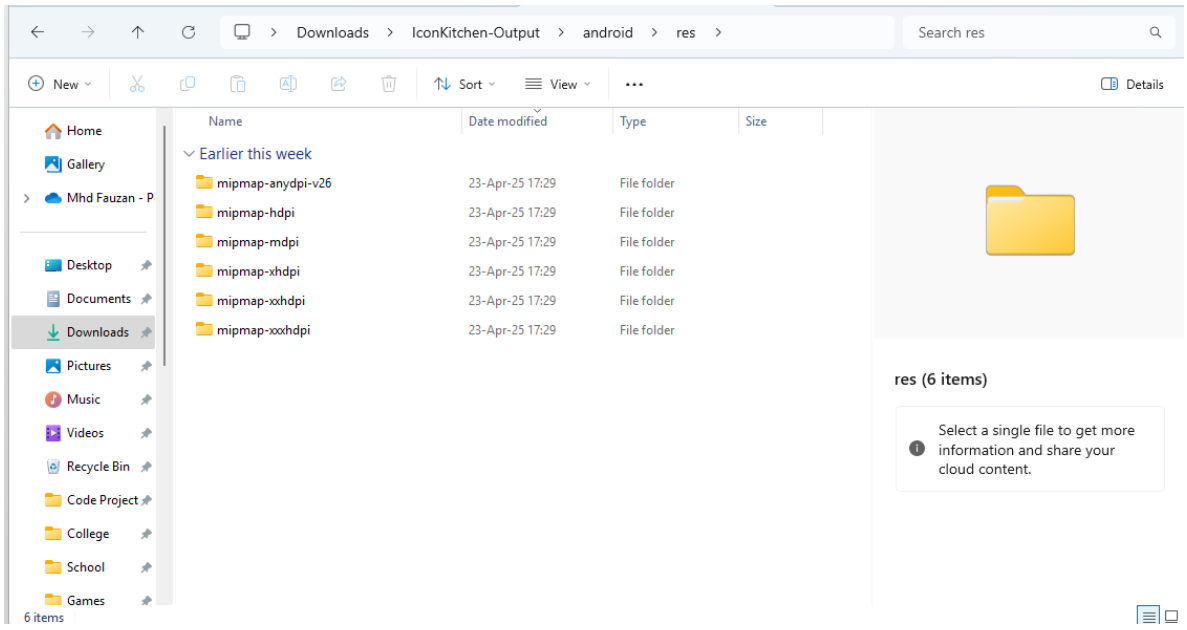
C. Pembahasan

Sebelum masuk ke pembahasan tiap kode, di sini saya ingin membahas tampilan secara umum terlebih dahulu. Desain yang saya buat memiliki beberapa perbedaan. Pertama saya menggunakan Rupiah, bukan Dollar. Karena saya rasa lebih sesuai dengan diri saya. Kedua, hasil perhitungan dibuat tebal dan besar. Sedangkan dari soal diletakan kecil dan tipis hampir tak terlihat. Padahal fungsi utamanya adalah melakukan perhitungan. Selain itu, pilihan tip dibuat secara ascending (yang rendah diletakan di paling atas) agar lebih intuitif.

Implementasi proyek kode ini menggunakan pendekatan MVVM dan Single Activity agar modular dan mudah dalam melakukan debugging.

Berikut beberapa dokumentasi saat membuat splash screen dan icon aplikasi menggunakan aplikasi web IconKitchen.





File-file diatas akan diletakkan ke folder `res` dari proyek ini.

1. MainActivity.kt:

Pertama, mendeklarasikan nama package-nya. Kemudian mengimpor librari yang dibutuhkan. Class `MainActivity` yang merupakan titik masuk utama aplikasi Android berbasis Jetpack Compose. Class ini meng-extend `ComponentActivity`, yang merupakan kelas dasar untuk aktivitas dalam Android yang mendukung Compose UI. Dalam method `onCreate()`, konten di-set melalui `setContent` yang membungkus seluruh UI dalam `TipCalculatorTheme` untuk menerapkan tema. `ViewModel` (`TipCalculatorViewModel`) diinisialisasi menggunakan `viewModel()` yang secara otomatis mengelola siklus hidupnya bersama Activity. Seluruh tampilan UI kemudian di-render melalui composable `TipCalculatorScreen` yang menerima instance `ViewModel` sebagai parameter.

Kemudian membuat fungsi `TipCalculatorPreview` yang merupakan composable khusus yang digunakan untuk keperluan pengembangan dan pengujian tampilan UI secara langsung di Android Studio. Dengan anotasi `@Preview`, komponen ini dapat menampilkan pratinjau (preview) dari `TipCalculatorScreen` tanpa perlu menjalankan aplikasi secara keseluruhan. Parameter `showBackground = true` memungkinkan latar belakang tema ditampilkan, sedangkan `showSystemUi = true` menambahkan simulasi status bar dan navigation bar perangkat untuk visualisasi yang lebih realistis (karena target device yang saya punya memiliki punch hole di tengah). Composable ini menggunakan `TipCalculatorTheme` untuk memastikan konsistensi tampilan dengan aplikasi utama, serta membuat instance `TipCalculatorViewModel` melalui `viewModel()` untuk menyediakan data dummy yang diperlukan selama proses desain.

2. TipCalculatorViewModel.kt:

Kelas `TipCalculatorViewModel` berperan sebagai komponen ViewModel yang mengelola logika bisnis dan state aplikasi berdasarkan pola MVVM. ViewModel ini menggunakan `MutableStateFlow` untuk menyimpan dan mengamati perubahan pada tiga state utama: jumlah tagihan (`billAmt`), rating layanan (`rating`), dan opsi pembulatan (`roundUp`). Setiap perubahan input pengguna ditangani melalui fungsi `onBillChanged`, `onRatingSelected`, dan `onRoundUpChanged` yang akan memperbarui state terkait. Fungsi `calculateTip()` melakukan perhitungan tip berdasarkan nilai tagihan dan persentase yang ditentukan dari rating (15% untuk Okay, 18% untuk Good, dan 20% untuk Amazing), dengan opsi pembulatan ke atas jika `roundUp` aktif.

3. TipCalculatorScreen.kt:

Kelas `TipCalculatorScreen.kt` memiliki fungsi utama menyusun tampilan menggunakan komponen Material Design 3 dalam struktur `Scaffold`.

Struktur `Scaffold` digunakan untuk membagi layar menjadi bagian-bagian penting, seperti `TopAppBar` untuk menampilkan judul aplikasi dan `content` yang berisi keseluruhan logika dan elemen interaktif.

Dalam fungsi `TipCalculatorScreen`, `viewModel` yang di-inject menggunakan `viewModel()` berfungsi sebagai penghubung antara UI dan logika bisnis. ViewModel ini menyediakan state seperti `bill`, `rating`, `tip`, dan `roundUp` melalui mekanisme `StateFlow`, yang kemudian dikonsumsi dalam UI menggunakan `collectAsState()`. Pendekatan ini menjamin bahwa UI akan selalu **reactive**, yaitu otomatis diperbarui saat ada perubahan data di ViewModel.

Elemen-elemen UI terdiri dari beberapa komponen kustom, yaitu `BillInputField` untuk memasukkan nominal tagihan, `TipRating` untuk memilih tingkat pelayanan, serta sebuah `Switch` untuk memilih apakah jumlah tip perlu dibulatkan atau tidak. Selain itu, terdapat tombol `Calculate` yang ketika ditekan akan memicu fungsi `calculateTip()` di ViewModel untuk menghitung jumlah tip berdasarkan input pengguna dan logika yang telah ditentukan.

Nilai tip yang telah dihitung kemudian diformat ke dalam format mata uang Rupiah menggunakan fungsi `formatAsCurrency`, yang menggunakan `DecimalFormat` untuk menambahkan pemisah ribuan dan dua digit desimal, kemudian ditampilkan dalam `Text` dengan gaya tipografi `headlineMedium` dan `fontWeight` tebal agar menonjol.

Selain fungsi utama, terdapat juga fungsi `@Preview` bernama `TipCalculatorScreenPreview` agar dapat melihat tampilan komponen di Android Studio tanpa perlu menjalankan aplikasinya. Di dalamnya, sebuah **fake ViewModel** disiapkan dengan input simulasi seperti tagihan sebesar **100** dan pilihan rating "**Amazing**" untuk membantu memvisualisasikan antarmuka dalam kondisi tertentu.

4. BillInputField.kt:

Komponen `BillInputField` merupakan komponen input khusus yang digunakan untuk menerima masukan nominal tagihan dari pengguna dalam bentuk teks. Komponen ini dirancang agar interaktif, responsif terhadap perubahan, serta memberikan pengalaman pengguna yang baik melalui format angka yang diformat secara otomatis. Di dalam komponen ini, nilai yang dimasukkan oleh pengguna disimpan dalam sebuah state lokal `formattedValue` menggunakan `remember` dan `mutableStateOf`. Nilai ini akan diperbarui secara otomatis setiap kali pengguna mengetik sesuatu di bidang input.

Bagian utama dari komponen ini adalah `OutlinedTextField`, yaitu komponen teks input dengan tampilan garis tepi khas Material Design. Komponen ini dikustomisasi dengan lebar penuh (`fillMaxWidth`) dan `padding horizontal` agar tata letaknya lebih proporsional. Properti `value` dari `OutlinedTextField` diikat ke `formattedValue`, sedangkan setiap perubahan nilai (`onValueChange`) akan diproses untuk membersihkan dan memformat input pengguna.

Logika pembersihan dan pemformatan dilakukan dengan tahap berikut. Pertama-tama, karakter selain angka dan titik (.) dihapus menggunakan ekspresi regular (**Regex**). Kemudian, untuk menghindari penggunaan titik ganda, titik yang berulang juga disederhanakan. Input selanjutnya dipisahkan menjadi bagian bilangan bulat dan desimal, di mana bagian desimal dibatasi maksimal dua digit yang merupakan **standar konvensi keuangan**. Untuk bagian bilangan bulat, `DecimalFormat` digunakan untuk menambahkan pemisah ribuan secara otomatis agar meningkatkan keterbacaan angka.

Pemformatan ini ditangani menggunakan blok `try-catch` untuk menghindari **crash** jika terjadi kesalahan konversi angka. Selanjutnya, nilai yang sudah diformat dikembalikan ke `formattedValue` untuk ditampilkan pada layar, dan juga dikirimkan kembali ke fungsi `onValueChange` agar dapat diproses oleh ViewModel di tingkat yang lebih tinggi.

`OutlinedTextField` juga dilengkapi dengan `KeyboardOptions` yang mengatur jenis papan ketik menjadi numerik (`KeyboardType.Number`) dan menyetel aksi **IME** (Input Method Editor) menjadi `Done`, sehingga pengalaman mengetik terasa lebih intuitif dan sesuai konteks. Untuk gaya teks, digunakan `TextStyle` dengan `fontWeight` tebal agar input terlihat lebih menonjol.

5. TipRating.kt:

Komponen `TipRating` adalah bagian dari antarmuka pengguna yang berfungsi untuk menangkap preferensi pengguna terhadap kualitas layanan yang diterima. Komponen ini disusun dengan menampilkan beberapa opsi penilaian berupa `radio button`. Masing-masing pilihan merepresentasikan tingkat pelayanan seperti "Okay (15%)", "Good (18%)", dan "Amazing (20%)", yang nantinya digunakan untuk menentukan persentase tip yang akan dihitung.

Komponen ini menggunakan `Column` sebagai wadah utama untuk menyusun elemen secara vertikal dan menggunakan modifier `fillMaxWidth()` agar mengisi seluruh lebar layar. Di dalamnya, pertama-tama terdapat `Text` yang menampilkan prompt atau pertanyaan kepada pengguna: "How was the service?".

Setiap opsi dalam daftar `options` di-loop menggunakan `forEach`, dan untuk setiap opsi, dibuat satu baris `Row` yang berisi `RadioButton` serta `Text` untuk labelnya. `Row` ini menggunakan properti `verticalAlignment = Alignment.CenterVertically` agar komponen radio dan teks berada pada satu garis horizontal secara rapi. `Radio button` akan tampak terpilih apabila opsi saat ini sama dengan nilai `selectedRating`, yang dikontrol dari luar komponen. Ketika pengguna memilih sebuah `radio button`, maka fungsi `onRatingSelected` akan dipanggil dengan nilai opsi tersebut, memungkinkan data tersebut diteruskan ke ViewModel untuk diproses lebih lanjut.

Komponen ini bersifat `stateless`, artinya ia tidak menyimpan state sendiri, tetapi menerima `selectedRating` dan `onRatingSelected` dari luar menggunakan prinsip **unidirectional data flow** yang mendorong pemisahan logika dan tampilan sehingga lebih mudah diuji, dipelihara, dan digunakan kembali di tempat lain.

6. AndroidManifest.kt:

File `AndroidManifest.xml` berperan sebagai konfigurasi utama dalam aplikasi Android. File ini mendeskripsikan komponen-komponen inti dari aplikasi, seperti aktivitas (activity), layanan (service), dan permissions yang dibutuhkan. Dalam proyek ini, manifest menetapkan satu aktivitas utama yaitu `MainActivity`, yang dideklarasikan sebagai titik masuk utama aplikasi melalui intent filter yang berisi aksi `android.intent.action.MAIN` dan kategori `android.intent.category.LAUNCHER`. Kombinasi ini memastikan bahwa `MainActivity` akan dijalankan pertama kali saat pengguna membuka aplikasi dari peluncur (launcher).

Properti `android:exported="true"` pada deklarasi aktivitas menunjukkan bahwa aktivitas ini dapat diakses dari luar aplikasi, yang merupakan persyaratan baru sejak Android 12 untuk aktivitas yang memiliki `intent-filter` dengan aksi `MAIN`. Atribut `android:label` dan `android:theme` digunakan untuk menetapkan judul tampilan dan tema awal aplikasi. Tema `Theme.App.Starting` digunakan baik dalam deklarasi aplikasi maupun aktivitas yang memilki **splash screen** modern.

Di tingkat aplikasi, terdapat beberapa properti penting seperti `android:allowBackup="true"` yang memungkinkan data aplikasi dicadangkan dan dipulihkan secara otomatis, serta `android:fullBackupContent` dan `android:dataExtractionRules` yang mengatur strategi pencadangan dan pemulihan data dalam format XML. Properti `android:supportsRtl="true"` memungkinkan aplikasi mendukung tata letak kanan-ke-kiri (RTL), penting untuk bahasa seperti Arab dan Ibrani. Selain itu, `android:icon` dan `android:roundIcon` menetapkan ikon aplikasi dalam versi biasa dan bundar.

Namespace tambahan tools digunakan untuk memberikan petunjuk atau informasi tambahan kepada Android Studio dan alat pengembang lainnya, seperti `tools:targetApi="31"` yang menunjukkan bahwa fitur tertentu hanya berlaku saat dijalankan pada API level 31 atau lebih tinggi.

7. themes.xml:

File `themes.xml` adalah bagian dari sistem styling Android yang mendefinisikan bagaimana tampilan aplikasi akan dirender secara keseluruhan. Dalam aplikasi Tip Calculator, file ini mendeklarasikan sebuah gaya bernama `Theme.TipCalculator` yang mewarisi dari tema dasar `android:Theme.Material.NoActionBar`. Dengan menggunakan `NoActionBar`, aplikasi tidak menampilkan action bar secara default, agar bisa menggunakan komponen `TopAppBar`.

8. styles.xml:

Style `Theme.App.Starting` dalam file styles.xml ini merupakan tema awal (launch theme) yang digunakan ketika aplikasi pertama kali dijalankan, dan memanfaatkan fitur **SplashScreen API** yang diperkenalkan sejak Android 12 (API 31). Tema ini mewarisi dari `Theme.SplashScreen`, yang secara otomatis menampilkan ikon dan latar belakang aplikasi saat proses peluncuran sedang berlangsung, memberikan transisi yang lebih mulus dan responsif sebelum masuk ke konten utama aplikasi.

Di dalam style ini, terdapat beberapa properti penting yang dikustomisasi. Pertama, `windowSplashScreenBackground` diatur ke warna `#ffebad`, yaitu warna **kuning pucat**. Selanjutnya, `windowSplashScreenAnimatedIcon` diatur ke `@mipmap/ic_launcher`, yaitu ikon utama aplikasi.

Durasi animasi splash ditentukan melalui `windowSplashScreenAnimationDuration` yang bernilai 200 milidetik. Setelah animasi splash selesai, sistem akan menerapkan `postSplashScreenTheme`, yang dalam hal ini mengarah ke `@style/Theme.TipCalculator`, yaitu tema utama aplikasi.

Terakhir, item `android:windowNoTitle` disetel ke `true` untuk memastikan bahwa **tidak ada judul jendela klasik Android** yang ditampilkan di bagian atas layar. Entah mengapa, meskipun style `Theme.TipCalculator` yang digunakan setelah splash screen telah mewarisi `android:Theme.Material.NoActionBar` yang secara default seharusnya **sudah menonaktifkan title bar**, efek tersebut tampaknya tidak berlaku secara konsisten saat dijalankan dalam konteks splash screen saat saya menjalankan aplikasinya. Oleh karena itu, atribut `android:windowNoTitle` **ditambahkan ulang** secara **eksplisit** di `Theme.App.Starting` untuk memastikan tampilan yang diharapkan benar-benar diterapkan.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/MinamotoYuki46/MeineStudienArbeit/tree/399c0e87b8948f8b3954fc6181785ffb21300adb/MobileDevelopment/Codex-Practicus/Modul%202