LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Farisa Adelia **NIM. 2110817120010**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Farisa Adelia
NIM : 2110817120010

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

MODUL 1 DICE ROLLER APP
SOAL 1

Buatlah sebuah aplikasi Dice Roller yang dapat menampilkan sebuah dadu yang dapat berubah-ubah tampilannya.

A. Source Code

```
1   package com.example.praktikumobile1
2
3   import androidx.appcompat.app.AppCompatActivity
4   import android.os.Bundle
5   import android.widget.Button
6   import android.widget.ImageView
7   import android.widget.TextView
8   import android.widget.Toast
9
10  /**
11   * activity ini untuk user menekan tombol roll tersebut
12   * on the screen.
13   */
14  class MainActivity : AppCompatActivity() {
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17          setContentView(R.layout.activity_main)
18          val rollButton: Button = findViewById(R.id.button)
19
20          rollButton.setOnClickListener { rollDice() }
21  //      di bawah ini untuk menampilkan dadu saat aplikasi pertama
22  kali di buka
23          rollDice()
24      }
25
26      //Roll the dice and update the screen with the result.
27      private fun rollDice() {
28          // Create new Dice object with 6 sides and roll it
29          val dice = Dice(6)
30          val diceRoll = dice.roll()
31          val diceRoll2 = dice.roll()
32
33
34          // Update the screen with the dice roll
35  //                          val   resultTextView:   TextView   =
36  findViewById(R.id.textView)
37          val diceImage: ImageView = findViewById(R.id.imageView)
```

```
38
39  //                      val   resultTextView2:   TextView   =
40  findViewById(R.id.textView2)
41          val          diceImage2:          ImageView          =
42  findViewById(R.id.imageView2)
43
44          // Determine which drawable resource ID to use based on
45  the dice roll
46
47          val drawableResource = when (diceRoll) {
48              1 -> R.drawable.dice_1
49              2 -> R.drawable.dice_2
50              3 -> R.drawable.dice_3
51              4 -> R.drawable.dice_4
52              5 -> R.drawable.dice_5
53              else -> R.drawable.dice_6
54          }
55          //  Update  the  ImageView  with  the  correct  drawable
56  resource ID
57
58          diceImage.setImageResource(drawableResource)
59          val drawableResource2 = when (diceRoll2) {
60              1 -> R.drawable.dice_1
61              2 -> R.drawable.dice_2
62              3 -> R.drawable.dice_3
63              4 -> R.drawable.dice_4
64              5 -> R.drawable.dice_5
65              else -> R.drawable.dice_6
66          }
67          //  Update  the  ImageView  with  the  correct  drawable
68  resource ID
69
70          diceImage2.setImageResource(drawableResource2)
71          // Update the content description
72
73          diceImage.contentDescription = diceRoll.toString()
74          diceImage2.contentDescription = diceRoll2.toString()
75
76          //  pengkondisian  terhadap  dadu  yang  sama  atau  beda
77  terhadap pesan yang muncul ke layar
78          if (diceRoll == diceRoll2) {
79              val toast = Toast.makeText(
80                  this,
81                  "Anda dapat dadu double yang sama!",
82                  Toast.LENGTH_SHORT
83              )
84              toast.show()
```

```
85          } else {
86              val toast2 = Toast.makeText(this, "Anda belum
87  beruntung!", Toast.LENGTH_SHORT)
88              toast2.show()
89          }
90      }
91
92  }
93
94
95  class Dice(private val numSides: Int) {
96
97      fun roll(): Int {
98          return (1..numSides).random()
99      }
100 }
```

Tabel 1. Source Code Activity

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       tools:context=".MainActivity">
9
10      <Button
11          android:id="@+id/button"
12          android:layout_width="wrap_content"
13          android:layout_height="wrap_content"
14          android:layout_marginTop="16dp"
15          android:text="@string/roll"
16          android:textSize="36sp"
17          app:layout_constraintEnd_toEndOf="parent"
18          app:layout_constraintStart_toStartOf="parent"
19          app:layout_constraintTop_toBottomOf="@+id/imageView" />
20
21      <ImageView
22          android:id="@+id/imageView"
23          android:layout_width="160dp"
24          android:layout_height="200dp"
25          app:layout_constraintBottom_toBottomOf="parent"
26          app:layout_constraintEnd_toEndOf="parent"
27          app:layout_constraintHorizontal_bias="0.498"
```

```
28          app:layout_constraintStart_toStartOf="parent"
29          app:layout_constraintTop_toTopOf="parent"
30          app:layout_constraintVertical_bias="0.499"
31          tools:srcCompat="@drawable/dice_1" />
32
33      <ImageView
34          android:id="@+id/imageView2"
35          android:layout_width="160dp"
36          android:layout_height="200dp"
37          app:layout_constraintBottom_toTopOf="@+id/imageView"
38          app:layout_constraintEnd_toEndOf="parent"
39          app:layout_constraintStart_toStartOf="parent"
40          app:layout_constraintTop_toTopOf="parent"
41          tools:srcCompat="@drawable/dice_1" />
42
43  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 2.Source Code Main

B. Output Program

Gambar 1. Output Soal 1

C. Pembahasan

Code MainActivity.kt tersebut adalah sebuah program yang berfungsi untuk menampilkan gambar dadu di layar dengan angka acak setiap kali tombol "Roll" ditekan. Program ini juga mengimplementasikan beberapa fitur tambahan seperti menampilkan pesan toast jika hasil dadu adalah double atau tidak.

Activity_amin.xml adalah file layout XML yang digunakan oleh activity MainActivity. Layout XML ini mendefinisikan tata letak tampilan yang akan ditampilkan oleh aplikasi padalayar.

MODUL 2 TIP CALCULATOR APP
SOAL 1

Buat aplikasi kalkulator tip dengan membangun tata letak terlebih dahulu, kemudian terapkan logika untuk menghitung tip berdasarkan input pengguna dan tambahkan tampilan yang menarik.

A. Source Code

```
1   package com.example.tiptime
2
3   import androidx.appcompat.app.AppCompatActivity
4   import android.os.Bundle
5   import android.os.PersistableBundle
6   import com.example.tiptime.databinding.ActivityMainBinding
7   import java.text.NumberFormat
8
9   class MainActivity : AppCompatActivity() {
10
11  //    Untuk mengambil referensi atribut yang dibuat pada file .xml
12      private lateinit var binding: ActivityMainBinding
13
14  //    Untuk menginisialisasi objek binding
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17
18          binding = ActivityMainBinding.inflate(layoutInflater)
19          setContentView(binding.root)
20
21  //        untuk memanggil function calculateTip pada calculateButton
    ketika diklik
22          binding.calculateButton.setOnClickListener { calculateTip() }
23      }
24
25  //    Function untuk memproses perhitungan tip
26      private fun calculateTip() {
27          val stringInTextField = binding.costOfService.text.toString()
28          val cost = stringInTextField.toDoubleOrNull()
29          if (cost == null) {
30              binding.tipResult.text = ""
31              return
32          }
33
    //    Untuk menghitung persenan
34          val tipPercentage = when
    (binding.tipOptions.checkedRadioButtonId) {
35              R.id.option_twenty_percent -> 0.20
36              R.id.option_eighteen_percent -> 0.18
37              else -> 0.15
38          }
39
40  //    Untuk membulatkan angka keatas ketika switch diaktifkan
41          var tip = tipPercentage * cost
42          if (binding.roundUpSwitch.isChecked) {
```

```
43              tip = kotlin.math.ceil(tip)
44          }
45
46  //    Untuk memformat nilai mata uang sesuai setting mobile user
47          val formattedTip =
    NumberFormat.getCurrencyInstance().format(tip)
48          binding.tipResult.text = getString(R.string.tip_amount,
    formattedTip)
49      }
50  }
51
```

Tabel 3. Source Code MainActivity.kt

Tabel 4. Source Code activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:padding="16dp"
9      tools:context=".MainActivity">
10
11 <!--  Untuk membuat kolom teks yang dapat diubah oleh user di dalam
12 aplikasi  -->
13     <EditText
14         android:id="@+id/cost_of_service"
15         android:layout_width="160dp"
16         android:layout_height="wrap_content"
17         android:hint="@string/cost_of_service"
18         android:inputType="numberDecimal"
19         app:layout_constraintStart_toStartOf="parent"
20         app:layout_constraintTop_toTopOf="parent" />
21
22 <!--  Untuk membuat teks yang tampil pada layar   -->
23     <TextView
24         android:id="@+id/service_question"
25         android:layout_width="wrap_content"
26         android:layout_height="wrap_content"
27         android:text="@string/how_was_the_service"
28         app:layout_constraintStart_toStartOf="parent"
29         app:layout_constraintTop_toBottomOf="@+id/cost_of_service"/>
30
31 <!--  Untuk mengelompokkan button yang dibuat menjadi 1 grup  -->
32     <RadioGroup
33         android:id="@+id/tip_options"
           android:layout_width="wrap_content"
34         android:layout_height="wrap_content"
35         android:checkedButton="@id/option_twenty_percent"
36         android:orientation="vertical"
37         app:layout_constraintStart_toStartOf="parent"
38         app:layout_constraintTop_toBottomOf="@id/service_question">
39
40         <!--  Tambahkan Radio Buttons disini  -->
```

```
41          <RadioButton
42              android:id="@+id/option_twenty_percent"
43              android:layout_width="wrap_content"
44              android:layout_height="wrap_content"
45              android:text="@string/amazing_20" />
46
47          <RadioButton
48              android:id="@+id/option_eighteen_percent"
49              android:layout_width="wrap_content"
50              android:layout_height="wrap_content"
51              android:text="@string/good_18" />
52
53          <RadioButton
54              android:id="@+id/option_fifteen_percent"
55              android:layout_width="wrap_content"
56              android:layout_height="wrap_content"
57              android:text="@string/okay_15" />
58
59      </RadioGroup>
60
61  <!--  Untuk membuat tombol switch  -->
62      <Switch
63          android:id="@+id/round_up_switch"
64          android:layout_width="0dp"
65          android:layout_height="wrap_content"
66          android:checked="true"
67          android:text="@string/round_up_tip"
68          app:layout_constraintEnd_toEndOf="parent"
69          app:layout_constraintStart_toStartOf="@+id/tip_options"
70          app:layout_constraintTop_toBottomOf="@+id/tip_options" />
71
72  <!--  Untuk membuat button berbentuk persegi  -->
73      <Button
74          android:id="@+id/calculate_button"
75          android:layout_width="0dp"
76          android:layout_height="wrap_content"
77          android:text="@string/calculate"
78          app:layout_constraintEnd_toEndOf="parent"
79          app:layout_constraintStart_toStartOf="parent"
80          app:layout_constraintTop_toBottomOf="@+id/round_up_switch" />
81
82      <TextView
83          android:id="@+id/tip_result"
84          android:layout_width="wrap_content"
85          android:layout_height="wrap_content"
86          app:layout_constraintEnd_toEndOf="parent"
87          app:layout_constraintTop_toBottomOf="@+id/calculate_button"
88          tools:text="Tip Amount: $10" />
89
90  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 5. Source Code build.gradle (:app)

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    namespace 'com.example.tiptime'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.tiptime"
        minSdk 24
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner
"androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        viewBinding = true
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation  'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.5.1'
}
```

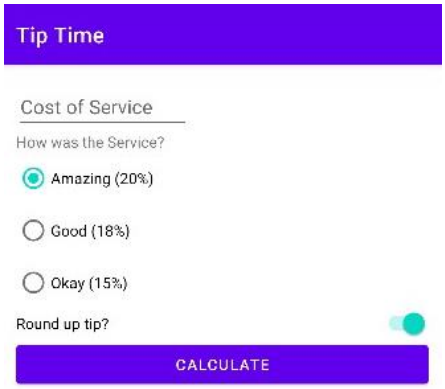
Tabel 6. Source Code strings.xml

```
1   <resources>
2       <string name="app_name">Tip Time</string>
3       <string name="tip_amount">Tip Amount : %s</string>
4       <string name="calculate">Calculate</string>
5       <string name="cost_of_service">Cost of Service</string>
6       <string name="how_was_the_service">How was the Service?</string>
7       <string name="amazing_20">Amazing (20%)</string>
8       <string name="good_18">Good (18%)</string>
9       <string name="okay_15">Okay (15%)</string>
10      <string name="round_up_tip">Round up tip?</string>
11  </resources>
```

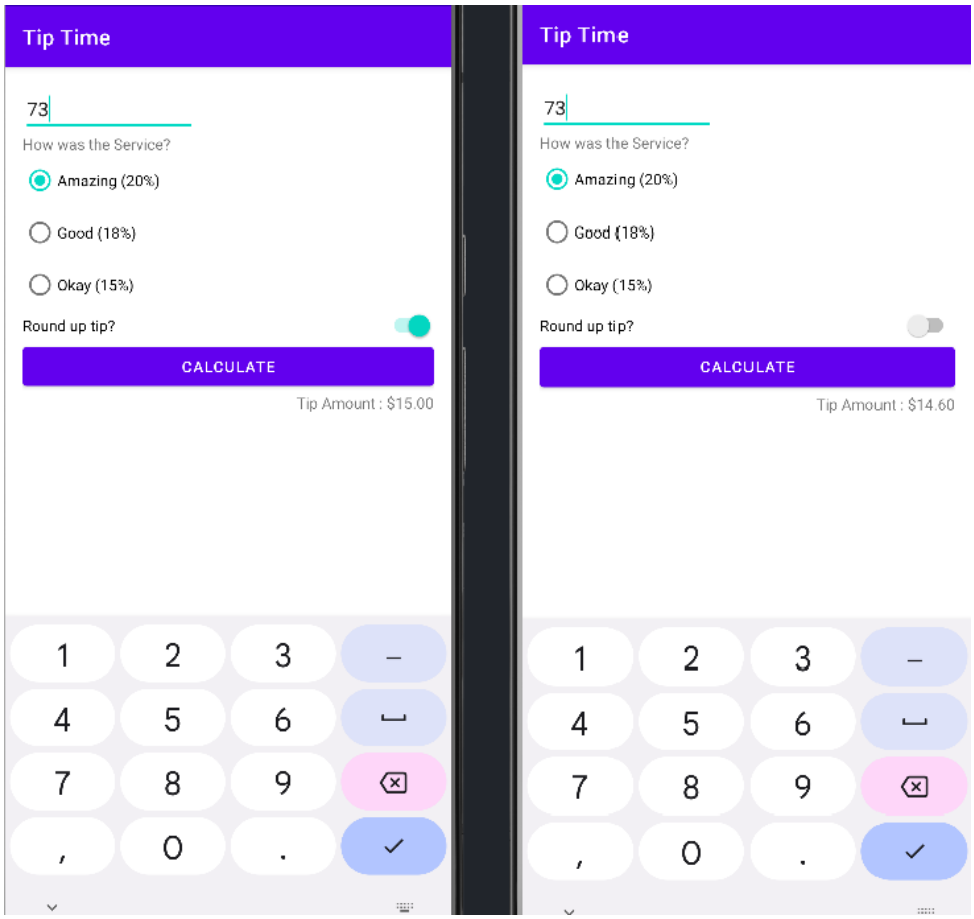
B. Output Program

Adapun unutk output dari program yang telah dibuat adalah sebagai berikut :

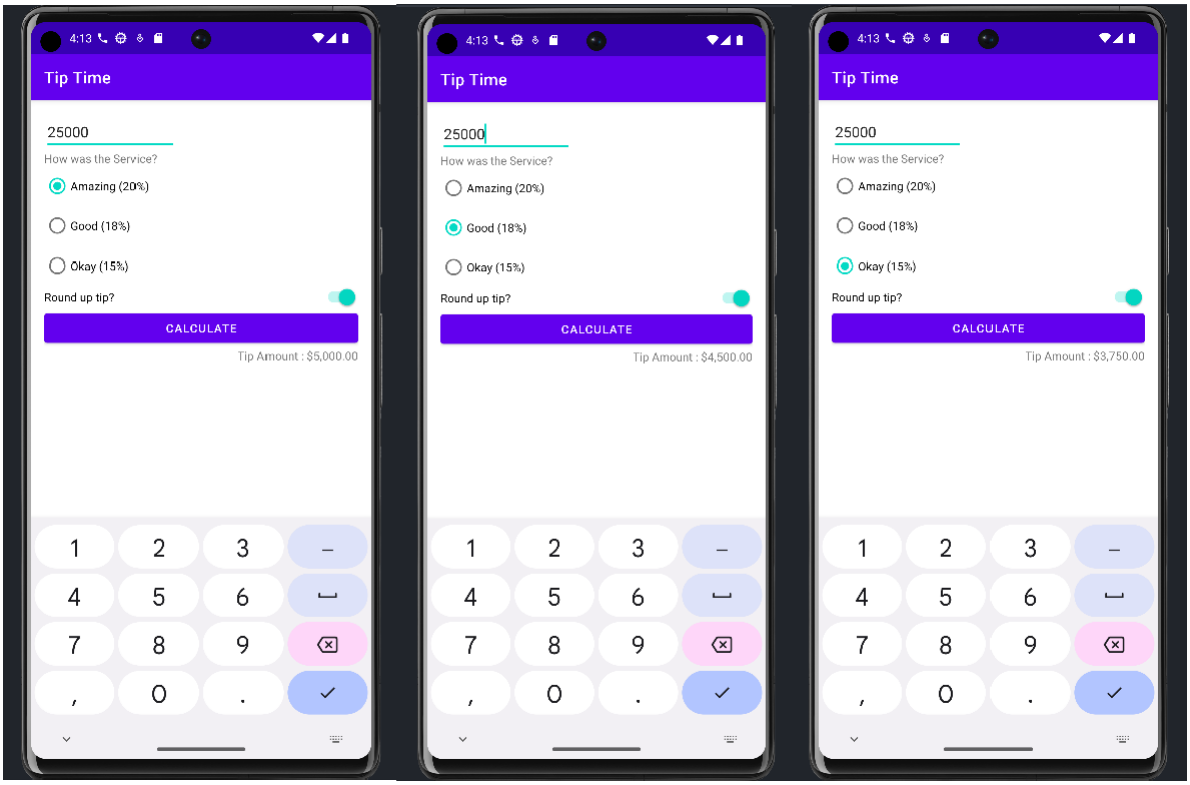
Gambar 2. Output Program Awal



Gambar 3. Calculate button clicked



Gambar 4. Output all radio button selected



C. Pembahasan

Pada praktikum ini membahas bagaimana membuat program calculator sederhana untuk menghitung uang tip. Program ini dijalankan dengan menggunakan bahasa kotlin dan Data Binding untuk menghubungkan layout XML pada Android Studio. Adapun penjelasan dari code-code yang ada pada program ini adalah :

MainActivity.kt

- Pada baris ke-3, dilakukan import library yang diperlukan untuk membangun aplikasi ini, termasuk **androidx.appcompat.app.AppCompatActivity** yang merupakan activity dasar dari aplikasi Android.
- Baris ke 4 - 7, merupakan deklarasi nama package dari aplikasi, dan import untuk library yang akan digunakan untuk activity.
- Baris ke 9 - 11, adalah deklarasi kelas MainActivity yang merupakan kelas dasar dari aplikasi.
- Baris ke 14 - 16, adalah deklarasi sebuah variabel **binding** dengan tipe **ActivityMainBinding**. Variabel ini digunakan untuk mengambil referensi atribut yang telah dibuat pada file XML.
- Baris ke 19 - 25, adalah fungsi **onCreate** yang digunakan untuk menginisialisasi objek binding dan mengatur layout activity yang akan digunakan pada aplikasi. Fungsi ini

juga menambahkan event listener pada button **calculateButton** yang akan memanggil fungsi **calculateTip()** ketika diklik.

- Baris ke 28 - 41, adalah fungsi **calculateTip()** yang digunakan untuk memproses perhitungan tip berdasarkan input dari pengguna seperti jumlah biaya dari service, persentase tip, dan pilihan untuk membulatkan tip ke atas.

Dalam fungsi **calculateTip()**, terdapat beberapa langkah yang dilakukan untuk memproses perhitungan tip, yaitu:

1. Mengambil input dari pengguna berupa jumlah biaya dari service pada text field **costOfService**.
2. Mengubah input tersebut menjadi tipe data **Double** dengan menggunakan fungsi **toDoubleOrNull()**.
3. Jika input dari pengguna tidak valid, maka output pada **tipResult** akan dikosongkan dan fungsi dihentikan menggunakan **return**.
4. Jika input dari pengguna valid, maka akan dihitung persentase tip berdasarkan opsi yang dipilih pada radio button **tipOptions**.
5. Jika opsi untuk membulatkan tip ke atas diaktifkan, maka tip akan dibulatkan ke atas menggunakan fungsi **kotlin.math.ceil()**.
6. Nilai tip yang dihasilkan akan diformat menggunakan **NumberFormat.getCurrencyInstance()** agar sesuai dengan format mata uang yang digunakan oleh pengguna.
7. Hasil perhitungan tip akan ditampilkan pada **tipResult** menggunakan fungsi **getString()**.

activity_main.xml :

1. EditText

Kolom teks yang terdapat di layout ini adalah EditText dengan id "cost_of_service". EditText ini digunakan untuk memasukkan angka yang merepresentasikan biaya pelayanan yang diterima.

2. TextView

Teks yang tampil pada layar adalah TextView dengan id "service_question". TextView ini memberikan pertanyaan kepada pengguna mengenai kualitas pelayanan yang diterima.

3. RadioGroup dan RadioButton

RadioGroup dan RadioButton digunakan untuk memberikan opsi tip yang dapat dipilih oleh pengguna. Terdapat 3 opsi tip yang disediakan, yaitu 15%, 18%, dan 20%.

4. Switch

Switch dengan id "round_up_switch" digunakan untuk membulatkan jumlah tip ke angka yang lebih besar jika diaktifkan oleh pengguna.

5. Button

Button dengan id "calculate_button" digunakan untuk menghitung jumlah tip berdasarkan nilai yang diinput oleh pengguna.

6. TextView

TextView dengan id "tip_result" digunakan untuk menampilkan hasil perhitungan jumlah tip yang diberikan kepada pelayan. Tampilan tip yang ditampilkan akan diformat ke dalam mata uang yang sesuai dengan setting mobile user.

MODUL 3 BUILD A SCROLLABLE LIST

SOAL 1

Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list

A. Source Code

Tabel 1. Source Code MainActivity.kt

```
/ MainActivity.kt
package com.example.modul3scrollablelist
import android.content.Intent import
android.net.Uri
import android.os.Bundle
import androidx.activity.ComponentActivity import
androidx.activity.compose.setContent import
androidx.compose.foundation.Image import
androidx.compose.foundation.background import
androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape import
androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items import
androidx.compose.material3.*
import androidx.compose.runtime.Composable import
androidx.compose.ui.Alignment import
androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip import
androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale import
androidx.compose.ui.res.painterResource import
androidx.compose.ui.text.font.FontWeight import
androidx.compose.ui.unit.dp
import androidx.navigation.NavController import
androidx.navigation.NavHostController import
androidx.navigation.compose.NavHost import
androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController import
com.example.modul3scrollablelist.ui.theme.Modul3ScrollableListTheme
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Modul3ScrollableListTheme {
                Surface(modifier = Modifier.fillMaxSize(), color =
MaterialTheme.colorScheme.background) {
                    MainScreen()
```

```
data class MovieItem( val id
Int,
        val title: String, val year:
        String, val plot: String,
        val imageResId: Int, val
        imdbLink: String
)
val sampleMovies = listOf(
MovieItem(1, "Pengabdi Setan 2: Communion", "2022",
"When the heavy storm hits, it wasn’t the storm that a family should fear but the
people
'non-human entities' who are out for them.",
 R.drawable.pengabdi, "https://www.imdb.com"),
MovieItem(2, "Siksa Kubur", "2024", "Tells about the punishment of the grave which
occurred after a man was buried.", R.drawable.siksa, "https://www.imdb.com"),
MovieItem(3, "Pengepungan di Bukit Duri", "2025",
"A special school for troubled children.
A teacher who is determined to discipline the students.
Here, teachers must not only teach, but survive the deadly attacks of their students.",
R.drawable.bukitduri, "https://www.imdb.com")
)


@Composable
fun MainScreen() {
        val navController = rememberNavController()
        NavHost(navController = navController, startDestination = "list")
{
                composable("list") { MovieListScreen(navController) }
                composable("detail/{movieId}") { backStackEntry ->
                        val movieId =
backStackEntry.arguments?.getString("movieId")?.toIntOrNull() val movie =
                        sampleMovies.find { it.id == movieId } if (movie != null)
                        DetailScreen(movie)
                }

        }

}
@Composable
fun MovieListScreen(navController: NavController) {
        LazyColumn(modifier = Modifier.padding(8.dp)) {
                items(sampleMovies) { movie -> Card(
                                shape = RoundedCornerShape(12.dp), modifier =
                                Modifier
                                        .padding(vertical = 8.dp)
                                        .fillMaxWidth()
                                        .background(MaterialTheme.colorScheme.surface)
```

```
                    ) {
                        Image(
                            painter = painterResource(id =
movie.imageResId),
                            contentDescription = movie.title,
                            contentScale = ContentScale.Crop,
                            modifier = Modifier
                                .fillMaxWidth()
                                .height(180.dp)
                                .clip(RoundedCornerShape(8.dp))
                        )
                        Spacer(modifier = Modifier.height(8.dp))
                        Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceBetween) {
                            Text(movie.title, fontWeight =
FontWeight.Bold)
                            Text(movie.year)
                        }
                        Spacer(modifier = Modifier.height(4.dp))
                        Text("Plot: ${movie.plot}", maxLines = 3)
                        Spacer(modifier = Modifier.height(8.dp))
                        Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceEvenly) {
                            Button(onClick = {
                                val intent = Intent(Intent.ACTION_VIEW,
Uri.parse(movie.imdbLink))

navController.context.startActivity(intent)
                            }) {
                                Text("IMDB")
                            }
                            Button(onClick = {

navController.navigate("detail/${movie.id}")
                            }) {
                                Text("Detail")
                            }
                        }
                    }
                }
            }
        }
}

@Composable
fun DetailScreen(movie: MovieItem) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
        Image(
            painter = painterResource(id = movie.imageResId),
            contentDescription = movie.title,
            contentScale = ContentScale.Crop,
            modifier = Modifier
                .fillMaxWidth()
```

```
                    .height(250.dp)
                    .clip(RoundedCornerShape(12.dp))
            )

            Spacer(modifier = Modifier.height(16.dp))
            Text(movie.title, style =
MaterialTheme.typography.titleLarge) Text("${movie.year}",
            style =
MaterialTheme.typography.labelMedium) Spacer(modifier =
            Modifier.height(8.dp)) Text("Plot:", fontWeight =
            FontWeight.Bold) Text(movie.plot)
    }
```

Tabel 2. Source Code AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher" android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round" android:supportsRtl="true"
        android:theme="@style/Theme.Modul3ScrollableList" tools:targetApi="31">
        <activity
            android:name=".MainActivity" android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.Modul3ScrollableList">
            <intent-filter>
                <action  android:name="android.intent.action.MAIN"  />


                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

B. Pembahasan

- Arsitektur Single Activity
- Aplikasi menggunakan NavController dan NavHost untuk menavigasi antar layar (daftar dan detail). Hal ini memungkinkan implementasi Single Activity Architecture, yang lebih ringan dan modern.

- LazyColumn
- LazyColumn digunakan untuk membuat list scrollable.
- Setiap elemen didesain dengan Card agar terlihat rapi dan profesional.
- Gambar diformat menggunakan ContentScale.Crop dan dibungkus dengan RoundedCornerShape.

- Intent Ekspilisit
- Tombol "IMDB" menerapkan explicit intent untuk membuka link IMDb menggunakan Intent.ACTION_VIEW.

- Navigasi ke Detail
- Tombol "Detail" menavigasi pengguna ke layar DetailScreen dengan navController.navigate("detail/${movie.id}").
- Data diambil berdasarkan movieId yang diparsing dari argumen navigasi.

- Layout Responsif
- Modifier seperti fillMaxWidth(), padding(), dan clip() digunakan agar tampilan tetap stabil dan menarik pada orientasi portrait maupun landscape.

- AndroidManifest.xml
- Sudah dikonfigurasi dengan benar untuk MainActivity.
- Menyediakan intent-filter untuk mendeklarasikan MAIN dan LAUNCHER.

SOAL 2

Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

JAWABAN : RecyclerView tetap digunakan karena fleksibel, stabil, dan memiliki ekosistem serta fitur lanjutan yang lebih matang, meskipun secara sintaksis lebih kompleks daripada LazyColumn di Jetpack Compose.

MODUL 4 VIEWMODEL AND DEBUGGING

SOAL 1

Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi

A. Source Code

Tabel 7. Source Code MainActivity.kt

```
package com.example.modul4viewmodelanddebugging

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.example.modul4viewmodelanddebugging.model.MovieItem
import
com.example.modul4viewmodelanddebugging.ui.theme.Modul3Scrollab
leListTheme
import
com.example.modul4viewmodelanddebugging.viewmodel.MovieViewMode
l
import
com.example.modul4viewmodelanddebugging.viewmodel.MovieViewMode
lFactory

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContent {
            Modul3ScrollableListTheme {
                Surface(modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colorScheme.background) {
                    val viewModel: MovieViewModel =
viewModel(factory = MovieViewModelFactory())
                    MainScreen(viewModel)
                }
            }
        }
    }
}

@Composable
fun MainScreen(viewModel: MovieViewModel) {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination =
"list") {
        composable("list") {
            MovieListScreen(navController, viewModel)
        }
        composable("detail") {
            val movie by
viewModel.selectedMovie.collectAsState()
            movie?.let {
                DetailScreen(it)
            }
        }
    }
}

@Composable
fun MovieListScreen(navController: NavController, viewModel:
MovieViewModel) {
    val context = LocalContext.current
    val movieList by viewModel.movies.collectAsState()

    LazyColumn(modifier = Modifier.padding(8.dp)) {
        items(movieList) { movie ->
            Card(
                shape = RoundedCornerShape(12.dp),
                modifier = Modifier
                    .padding(vertical = 8.dp)
                    .fillMaxWidth()
            ) {
                Column(modifier = Modifier.padding(12.dp)) {
                    Image(
                        painter = painterResource(id =
movie.imageResId),
                        contentDescription = movie.title,
                        contentScale = ContentScale.Crop,
                        modifier = Modifier
```

```
                                .fillMaxWidth()
                                .height(180.dp)
                                .clip(RoundedCornerShape(8.dp))
                        )
                        Spacer(modifier = Modifier.height(8.dp))
                        Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceBetween) {
                            Text(movie.title, fontWeight =
FontWeight.Bold)
                            Text(movie.year)
                        }
                        Spacer(modifier = Modifier.height(4.dp))
                        Text("Plot: ${movie.plot}", maxLines = 3)
                        Spacer(modifier = Modifier.height(8.dp))
                        Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceEvenly) {
                            Button(onClick = {
                                viewModel.onImdbClick(movie)
                                val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(movie.imdbLink))
                                context.startActivity(intent)
                            }) {
                                Text("IMDB")
                            }
                            Button(onClick = {
                                viewModel.selectMovie(movie)
                                viewModel.onDetailClick(movie)
                                navController.navigate("detail")
                            }) {
                                Text("Detail")
                            }
                        }
                    }
                }
            }
        }
    }
}

@Composable
fun DetailScreen(movie: MovieItem) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
        Image(
            painter = painterResource(id = movie.imageResId),
            contentDescription = movie.title,
            contentScale = ContentScale.Crop,
            modifier = Modifier
                .fillMaxWidth()
                .height(250.dp)
```

```
                .clip(RoundedCornerShape(12.dp))
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(movie.title, style =
MaterialTheme.typography.titleLarge)
        Text(movie.year, style =
MaterialTheme.typography.labelMedium)
        Spacer(modifier = Modifier.height(8.dp))
        Text("Plot:", fontWeight = FontWeight.Bold)
        Text(movie.plot)
    }
}
```

Tabel 8. Source Code AndroidManifest.xml

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

    <application
        android:allowBackup="true"

android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Modul3ScrollableList"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.Modul3ScrollableList">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Tabel 9. Source Code MovieItem.kt

```
package com.example.modul4viewmodelanddebugging.model

data class MovieItem(
    val id: Int,
    val title: String,
    val year: String,
    val plot: String,
    val imageResId: Int,
    val imdbLink: String
)
```

Tabel 10. Source Code MovieViewModel.kt

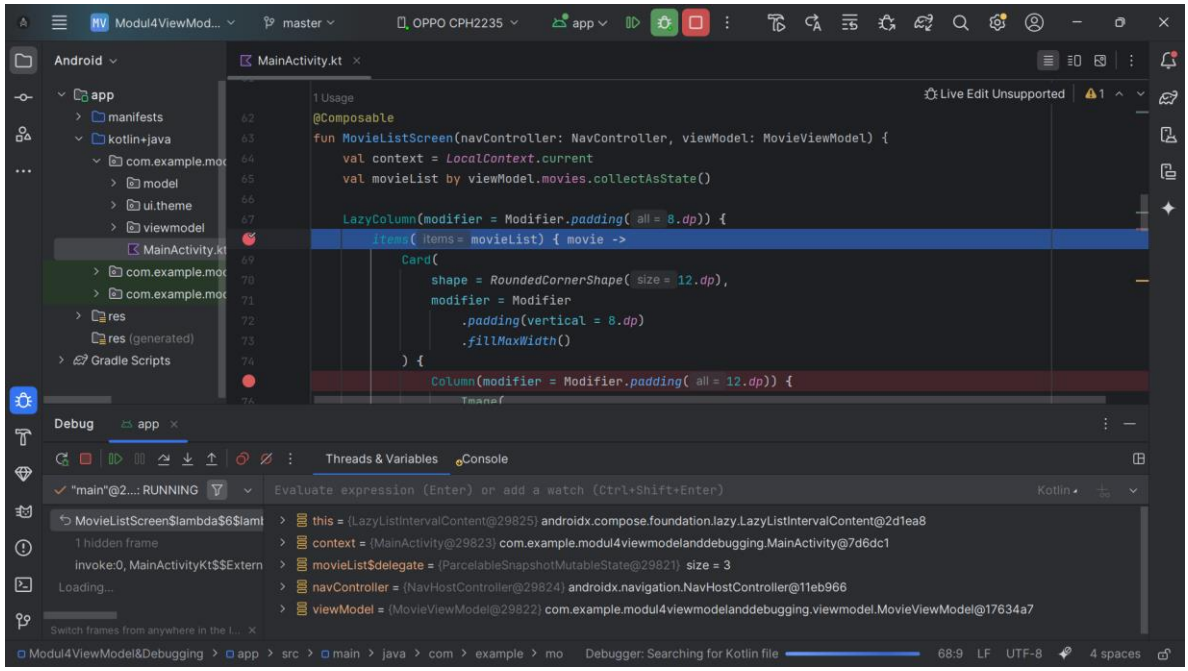
```
package com.example.modul4viewmodelanddebugging.viewmodel

import android.util.Log
import androidx.lifecycle.ViewModel
import com.example.modul4viewmodelanddebugging.R
import com.example.modul4viewmodelanddebugging.model.MovieItem
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow

class MovieViewModel : ViewModel() {

    private val _movies =
MutableStateFlow<List<MovieItem>>(emptyList())
    val movies: StateFlow<List<MovieItem>> = _movies

    private val _selectedMovie =
MutableStateFlow<MovieItem?>(null)
    val selectedMovie: StateFlow<MovieItem?> = _selectedMovie

    init {
        val sample = listOf(
            MovieItem(1, "Pengabdi Setan 2: Communion", "2022",
"When the heavy storm hits...", R.drawable.pengabdi,
"https://www.imdb.com"),
            MovieItem(2, "Siksa Kubur", "2024", "Tells about
the punishment of the grave...", R.drawable.siksa,
"https://www.imdb.com"),
            MovieItem(3, "Pengepungan di Bukit Duri", "2025",
"A special school for troubled children...",
R.drawable.bukitduri, "https://www.imdb.com")
        )

        Log.d("MovieViewModel", "Data masuk ke dalam list:
${sample.size} item")
        _movies.value = sample
    }
```

```
    fun selectMovie(movie: MovieItem) {
        Log.d("MovieViewModel", "Data yang dipilih untuk
detail: $movie")
        _selectedMovie.value = movie
    }

    fun onImdbClick(movie: MovieItem) {
        Log.d("MovieViewModel", "Tombol IMDB ditekan:
${movie.title}")
    }

    fun onDetailClick(movie: MovieItem) {
        Log.d("MovieViewModel", "Tombol Detail ditekan:
${movie.title}")
    }
}
```

Tabel 11. Source Code MovieViewModelFactory.kt

```
package com.example.modul4viewmodelanddebugging.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class MovieViewModelFactory : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>):
T {
        return MovieViewModel() as T
    }
}
```

B. Output Program



Gambar 5. Screenshoot Hasil Jawaban Soal 1

Siksa Kubur

2024

Plot:

Tells about the punishment of the grave...

Gambar 6. Screenshoot Hasil Jawaban Soal 1

Gambar 7. Screenshoot Hasil Jawaban Soal 1

C. Pembahasan

- ViewModel untuk Pengolahan Data
  - ViewModel digunakan sebagai komponen arsitektural untuk menyimpan dan mengelola data list item
  - Pendekatan ini mengikuti prinsip MVVM (Model-View-ViewModel) dan memastikan data tetap terjaga meskipun konfigurasi perangkat berubah (misalnya rotasi layar).
  - Data tidak disimpan di Activity/Fragment, melainkan di ViewModel yang bersifat lifecycle-aware.
- ViewModelFactory
  - ViewModel dibuat menggunakan ViewModelProvider.Factory untuk menyuplai dependensi jika dibutuhkan (misalnya repository atau argumen).
  - Hal ini membantu ketika ViewModel memiliki konstruktor non-default.
- StateFlow untuk State Management
  - StateFlow digunakan untuk mengelola event onClick dan list data item dari ViewModel ke Fragment.
  - Dibanding LiveData, StateFlow menawarkan pendekatan reaktif dan mendukung coroutine secara penuh.
  - StateFlow digunakan dengan collectAsState() dalam Jetpack Compose agar UI otomatis merespons perubahan data.
- Logging
  - Logging ditambahkan untuk membantu proses debugging dan pemantauan perilaku aplikasi
  - Saat data item ditambahkan ke list

    ```
    Log.d("MovieViewModel", "Item ditambahkan:
    ${movie.title}")
    ```
  - Saat tombol “Detail” dan “Explicit Intent” ditekan

    ```
    Log.d("MovieListScreen", "Tombol Detail ditekan untuk
    id: ${movie.id}")
    Log.d("MovieListScreen", "Explicit Intent ditekan:
    ${movie.imdbUrl}")
    ```

- Saat berpindah ke halaman Detail

  ```
  Log.d("DetailScreen", "Menampilkan data untuk id: $movieId")
  ```

- Debugging Menggunakan Debugger

  Debugger di Android Studio digunakan untuk menganalisis perilaku aplikasi saat runtime:

  - Breakpoint ditempatkan pada baris pemrosesan intent navigasi untuk memantau data yang diparsing.
  - Step Into: Masuk ke dalam fungsi yang sedang dipanggil, berguna untuk memeriksa logika internal.
  - Step Over: Melewati fungsi tanpa masuk ke dalamnya, melanjutkan eksekusi ke baris berikutnya.
  - Step Out: Keluar dari fungsi saat ini dan kembali ke pemanggil sebelumnya.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

JAWABAN : Application class adalah kelas dasar yang menggambarkan seluruh siklus dari sebuah aplikasi. Class ini hanya dibuat sekali saat aplikasi dijalankan dan bertahan selama aplikasi masih aktif di memori. Fungsi utama dari Application Class adalah sebagai tempat untuk melakukan inisialisasi yang hanya dilakukan sekali, menyediakan akses global ke aplikasi, menjadi entry poin untuk konfigurasi aplikasi, dan dapat memonitoring seperti menangkap error atau mengawasi aplikasi secara menyeluruh.

MODUL 5
CONNECT TO THE INTERNET

SOAL 1

Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi

A. Source Code

Tabel 12. Source Code MainActivity.kt

```
package com.example.modul5connecttotheinternetcopy

import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.core.net.toUri
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import androidx.navigation.compose.NavHost
```

```
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import
com.example.modul5connecttotheinternetcopy.model.MovieItem
import
com.example.modul5connecttotheinternetcopy.ui.theme.Modul3Scrol
lableListTheme
import
com.example.modul5connecttotheinternetcopy.viewmodel.MovieViewM
odel
import
com.example.modul5connecttotheinternetcopy.viewmodel.MovieViewM
odelFactory
import
com.example.modul5connecttotheinternetcopy.viewmodel.UserViewMo
del
import
com.example.modul5connecttotheinternetcopy.viewmodel.UserViewMo
delFactory


class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Modul3ScrollableListTheme {
                Surface(modifier = Modifier.fillMaxSize(),
color = MaterialTheme.colorScheme.background) {
                    val movieViewModel: MovieViewModel =
viewModel(factory = MovieViewModelFactory())
                    val userViewModel: UserViewModel =
viewModel(factory = UserViewModelFactory(application))
                    MainScreen(movieViewModel, userViewModel)
                }
            }
        }
    }
}
@Composable
fun ProfileScreen(userViewModel: UserViewModel) {
    val user by userViewModel.user.collectAsState()
    val context = LocalContext.current

    LaunchedEffect(Unit) {
        userViewModel.fetchUser(true)
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
```

```
        Text("Profile", style =
MaterialTheme.typography.titleLarge)
        Spacer(modifier = Modifier.height(16.dp))

        if (user != null) {
            Text("Name: ${user?.name?.title}
${user?.name?.first} ${user?.name?.last}")
            Text("Email: ${user?.email}")
            Text("Phone: ${user?.phone}")
            Text("Cell: ${user?.cell}")
            Text("Address: ${user?.location?.street?.number}
${user?.location?.street?.name}, ${user?.location?.city},
${user?.location?.state}, ${user?.location?.country},
${user?.location?.postcode}")
        } else {
            Text("Loading user data or no data available
offline...")
        }

        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = {
            userViewModel.fetchUser(true)
        }) {
            Text("Refresh User")
        }
        Spacer(modifier = Modifier.height(8.dp))
        Button(onClick = {  }) {
            Text("Logout")
        }
    }
}

@Composable
fun MainScreen(
    viewModel: MovieViewModel,
    userViewModel: UserViewModel
) {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination =
"list") {
        composable("list") {
            MovieListScreen(navController, viewModel)
        }
        composable("detail") {
            val movie by
viewModel.selectedMovie.collectAsState()
            movie?.let {
                DetailScreen(it)
            }
        }
        composable("profile") {
            ProfileScreen(userViewModel)
```

```
        }
    }
}


@Composable
fun MovieListScreen(navController: NavController, viewModel:
MovieViewModel) {
    val context = LocalContext.current
    val movieList by viewModel.movies.collectAsState()

    Column(modifier = Modifier.fillMaxSize()) {
        Button(
            onClick = { navController.navigate("profile") },
            modifier = Modifier
                .padding(8.dp)
                .fillMaxWidth()
        ) {
            Text("Go to Profile")
        }

        LazyColumn(modifier =
Modifier.padding(8.dp).weight(1f)) {
            items(movieList) { movie ->
                Card(
                    shape = RoundedCornerShape(12.dp),
                    modifier = Modifier
                        .padding(vertical = 8.dp)
                        .fillMaxWidth()
                ) {
                    Column(modifier = Modifier.padding(12.dp))
{
                        Image(
                            painter = painterResource(id =
movie.imageResId),
                            contentDescription = movie.title,
                            contentScale = ContentScale.Crop,
                            modifier = Modifier
                                .fillMaxWidth()
                                .height(180.dp)
                                .clip(RoundedCornerShape(8.dp))
                        )
                        Spacer(modifier =
Modifier.height(8.dp))
                        Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceBetween) {
                            Text(movie.title, fontWeight =
FontWeight.Bold)
                            Text(movie.year)
                        }
                        Spacer(modifier =
Modifier.height(4.dp))
```

```
                            Text("Plot: ${movie.plot}", maxLines =
3)
                            Spacer(modifier =
Modifier.height(8.dp))
                            Row(Modifier.fillMaxWidth(),
horizontalArrangement = Arrangement.SpaceEvenly) {
                                Button(onClick = {
                                    viewModel.onImdbClick(movie)
                                    val intent =
Intent(Intent.ACTION_VIEW, movie.imdbLink.toUri())
                                    context.startActivity(intent)
                                }) {
                                    Text("IMDB")
                                }
                                Button(onClick = {
                                    viewModel.selectMovie(movie)
                                    viewModel.onDetailClick(movie)

navController.navigate("detail")
                                }) {
                                    Text("Detail")
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}


@Composable
fun DetailScreen(movie: MovieItem) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
    ) {
        Image(
            painter = painterResource(id = movie.imageResId),
            contentDescription = movie.title,
            contentScale = ContentScale.Crop,
            modifier = Modifier
                .fillMaxWidth()
                .height(250.dp)
                .clip(RoundedCornerShape(12.dp))
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(movie.title, style =
MaterialTheme.typography.titleLarge)
        Text(movie.year, style =
MaterialTheme.typography.labelMedium)
```

```
        Spacer(modifier = Modifier.height(8.dp))
        Text("Plot:", fontWeight = FontWeight.Bold)
        Text(movie.plot)
    }
}
```

Tabel 13. Source Code AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET"
/>

    <application
        android:allowBackup="true"

android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Modul3ScrollableList"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.Modul3ScrollableList">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Tabel 14. Source Code MovieItem.kt

```
package com.example.modul4viewmodelanddebugging.model

data class MovieItem(
    val id: Int,
```

```
    val title: String,
    val year: String,
    val plot: String,
    val imageResId: Int,
    val imdbLink: String
)
```

Tabel 15. Source Code MovieViewModel.kt

```
package com.example.modul4viewmodelanddebugging.viewmodel

import android.util.Log
import androidx.lifecycle.ViewModel
import com.example.modul4viewmodelanddebugging.R
import com.example.modul4viewmodelanddebugging.model.MovieItem
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow

class MovieViewModel : ViewModel() {

    private val _movies =
MutableStateFlow<List<MovieItem>>(emptyList())
    val movies: StateFlow<List<MovieItem>> = _movies

    private val _selectedMovie =
MutableStateFlow<MovieItem?>(null)
    val selectedMovie: StateFlow<MovieItem?> = _selectedMovie

    init {
        val sample = listOf(
            MovieItem(1, "Pengabdi Setan 2: Communion", "2022",
"When the heavy storm hits...", R.drawable.pengabdi,
"https://www.imdb.com"),
            MovieItem(2, "Siksa Kubur", "2024", "Tells about
the punishment of the grave...", R.drawable.siksa,
"https://www.imdb.com"),
            MovieItem(3, "Pengepungan di Bukit Duri", "2025",
"A special school for troubled children...",
R.drawable.bukitduri, "https://www.imdb.com")
        )

        Log.d("MovieViewModel", "Data masuk ke dalam list:
${sample.size} item")
        _movies.value = sample
    }

    fun selectMovie(movie: MovieItem) {
        Log.d("MovieViewModel", "Data yang dipilih untuk
detail: $movie")
        _selectedMovie.value = movie
    }
```

```
    fun onImdbClick(movie: MovieItem) {
        Log.d("MovieViewModel", "Tombol IMDB ditekan:
${movie.title}")
    }

    fun onDetailClick(movie: MovieItem) {
        Log.d("MovieViewModel", "Tombol Detail ditekan:
${movie.title}")
    }
}
```

Tabel 16. Source Code MovieViewModelFactory.kt

```
package com.example.modul4viewmodelanddebugging.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class MovieViewModelFactory : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>):
T {
        return MovieViewModel() as T
    }
}
```

Tabel 17. Source Code User.kt

```
package com.example.modul5connecttotheinternetcopy.model

import androidx.room.Embedded
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.serialization.Serializable

@Serializable
@Entity(tableName = "users")
data class User(
    @PrimaryKey val email: String,
    val gender: String,
    @Embedded val name: Name,
    @Embedded val location: Location,
    val phone: String,
    val cell: String,
    @Embedded val picture: Picture
)

@Serializable
data class Name(
    val title: String,
```

```
    val first: String,
    val last: String
)

@Serializable
data class Location(
    @Embedded val street: Street,
    val city: String,
    val state: String,
    val country: String,
    val postcode: Int
)

@Serializable
data class Street(
    val number: Int,
    val name: String
)

@Serializable
data class Picture(
    val large: String,
    val medium: String,
    val thumbnail: String
)
```

Tabel 18. Source Code UserResponse.kt

```
package com.example.modul5connecttotheinternetcopy.model

import com.example.modul5connecttotheinternetcopy.model.User
import kotlinx.serialization.Serializable

@Serializable
data class UserResponse(
    val results: List<User>
)
```

Tabel 19. Source Code UserRepository.kt

```
package com.example.modul5connecttotheinternetcopy.data

import com.example.modul5connecttotheinternetcopy.model.User
import com.example.modul5connecttotheinternetcopy.ApiService
import com.example.modul5connecttotheinternetcopy.data.UserDao
import kotlinx.coroutines.flow.Flow

class UserRepository(private val apiService: ApiService,
private val userDao: UserDao) {

```

```
    fun getUser(): Flow<User?> {
        return userDao.getUser()
    }

    suspend fun refreshUser() {
        try {
            val response = apiService.getRandomUser()
            val user = response.results.firstOrNull()
            if (user != null) {
                userDao.deleteAllUsers()
                userDao.insertUser(user)
            }
        } catch (e: Exception) {
            println("Error refreshing user: ${e.message}")
            throw e
        }
    }
}
```

Tabel 20. Source Code UserViewModel.kt

```
package com.example.modul5connecttotheinternetcopy.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import
com.example.modul5connecttotheinternetcopy.data.UserRepository
import com.example.modul5connecttotheinternetcopy.model.User
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.stateIn
import kotlinx.coroutines.flow.SharingStarted
import kotlinx.coroutines.launch

class UserViewModel(private val repository: UserRepository) :
ViewModel() {
    val user: StateFlow<User?> = repository.getUser()
        .stateIn(viewModelScope,
SharingStarted.WhileSubscribed(5000), null)

    init {
        fetchUser(true)
    }

    fun fetchUser(forceRefresh: Boolean = false) {
        viewModelScope.launch {
            if (forceRefresh) {
                try {
                    repository.refreshUser()
                } catch (e: Exception) {
                    println("Error fetching user from network:
${e.message}")
```

```
                    }
                }
            }
        }
}
```

Tabel 21. Source Code UserViewModelFactory.kt

```
package com.example.modul5connecttotheinternetcopy.viewmodel

import android.app.Application
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import
com.example.modul5connecttotheinternetcopy.data.AppDatabase
import
com.example.modul5connecttotheinternetcopy.RetrofitInstance
import
com.example.modul5connecttotheinternetcopy.data.UserRepository
import
com.example.modul5connecttotheinternetcopy.viewmodel.UserViewMo
del

class UserViewModelFactory(private val application:
Application) : ViewModelProvider.Factory {
    @Suppress("UNCHECKED_CAST")
    override fun <T : ViewModel> create(modelClass: Class<T>):
T {
        if
(modelClass.isAssignableFrom(UserViewModel::class.java)) {
            val database = AppDatabase.getDatabase(application)
            val userDao = database.userDao()
            val apiService = RetrofitInstance.api
            val repository = UserRepository(apiService,
userDao)
            return UserViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel
class")
    }
}
```

Tabel 22. Source Code ApiService

```
package com.example.modul5connecttotheinternetcopy

import
com.example.modul5connecttotheinternetcopy.model.UserResponse
import retrofit2.http.GET
```

```
interface ApiService {
    @GET("api/")
    suspend fun getRandomUser(): UserResponse
}
```

Tabel 23. AppDatabase

```
package com.example.modul5connecttotheinternetcopy.data

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import com.example.modul5connecttotheinternetcopy.model.User
import com.example.modul5connecttotheinternetcopy.data.UserDao

@Database(entities = [User::class], version = 1, exportSchema =
false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "app_database"
                ).build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

Tabel 24. Source Code RetrofitInstance

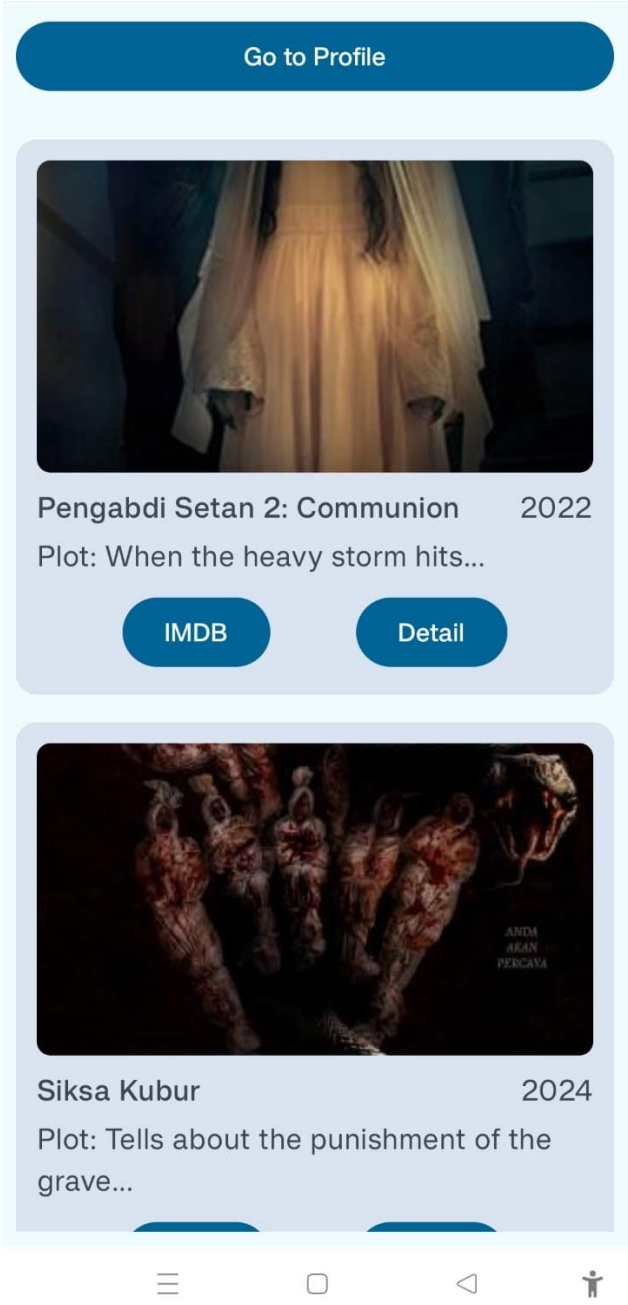
```
package com.example.modul5connecttotheinternetcopy

import
com.jakewharton.retrofit2.converter.kotlinx.serialization.asCo
nverterFactory
import kotlinx.serialization.json.Json
import retrofit2.Retrofit
import okhttp3.MediaType.Companion.toMediaType
```

```
object RetrofitInstance {
    private val json = Json { ignoreUnknownKeys = true }

    private val retrofit by lazy {
        Retrofit.Builder()
            .baseUrl("https://randomuser.me/")

.addConverterFactory(json.asConverterFactory("application/json
".toMediaType()))
            .build()
    }

    val api: ApiService by lazy {
        retrofit.create(ApiService::class.java)
    }
}
```

Tabel 25. Source Code UserDao

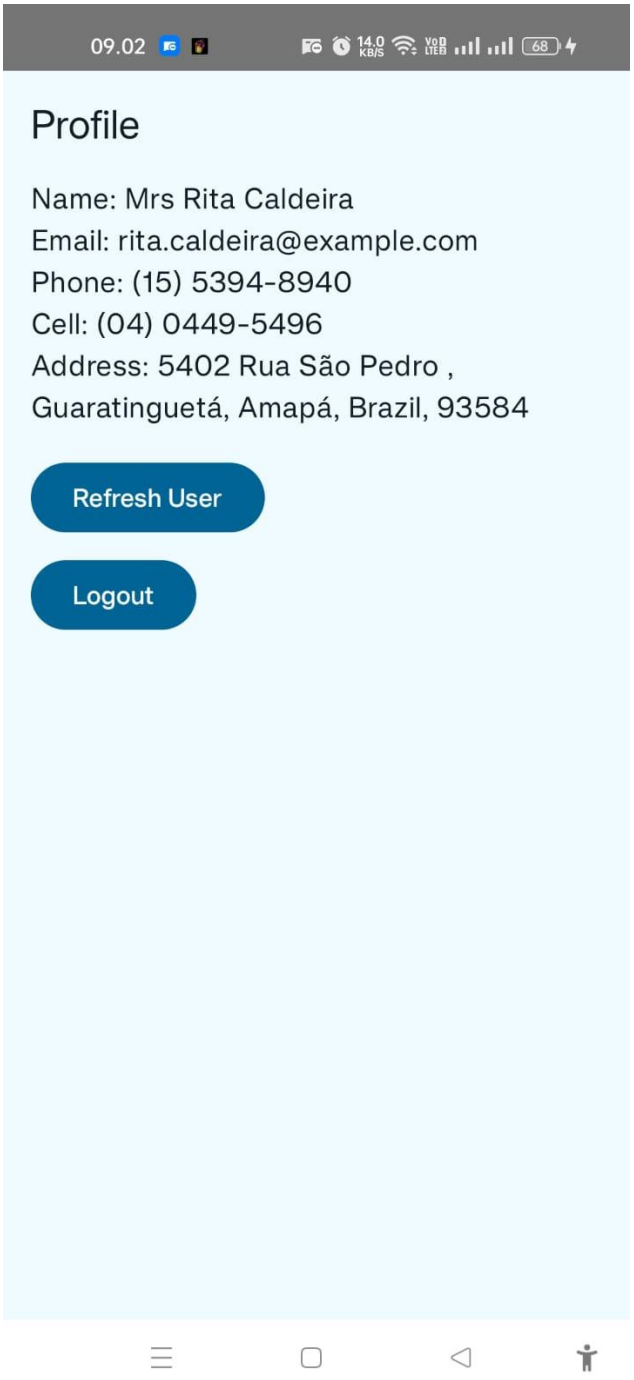
```
package com.example.modul5connecttotheinternetcopy.data

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.example.modul5connecttotheinternetcopy.model.User
import kotlinx.coroutines.flow.Flow

@Dao
interface UserDao {
    @Query("SELECT * FROM users LIMIT 1")
    fun getUser(): Flow<User?>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)

    @Query("DELETE FROM users")
    suspend fun deleteAllUsers()
}
```

B. Output Program



Gambar 8. Screenshot Hasil Soal 1

Gambar 9. Screenshot Hasil Soal 1



Gambar 10. Screenshot Hasil Soal 1

C. Pembahasan

- Networking dengan Retrofit dan KotlinX Serialization
  - Aplikasi menggunakan Retrofit sebagai library utama untuk mengambil data dari remote API.
  - Untuk parsing JSON, digunakan KotlinX Serialization dengan konfigurasi converter di Retrofit:

```
val retrofit = Retrofit.Builder()
    .baseUrl("https://randomuser.me/")
addConverterFactory(json.asConverterFactory("application
/json".toMediaType()))
    .build()
```

  - Struktur response dibungkus dalam generic response handler untuk menampilkan status loading, success, dan error.
  - Pengambilan data dari ViewModel dilakukan melalui Flow untuk streaming data: `val userList: StateFlow<ApiResponse<List<User>>> = _userList`
- Image Loading dengan Coil
  - Aplikasi menggunakan library Coil untuk memuat gambar profil pengguna yang diambil dari API.
  - Coil diintegrasikan langsung di Jetpack Compose:

```
Image(
    painter = rememberImagePainter(data =
user.picture.large),
    contentDescription = null,
    modifier = Modifier.size(64.dp).clip(CircleShape),
    contentScale = ContentScale.Crop
)
```

- Navigasi ke Detail
  - Navigasi menggunakan NavController menuju halaman detail dengan menyertakan ID atau parameter unik pengguna
  - Detail pengguna diambil berdasarkan parameter tersebut dari repository atau local database.
  - StateFlow digunakan dengan collectAsState() dalam Jetpack Compose agar UI otomatis merespons perubahan data.
- Data Persistence

    - Aplikasi menerapkan offline-first strategy dengan menyimpan data dari API ke Room database.
    - Jika API tidak dapat dijangkau, data tetap ditampilkan dari cache lokal.
- Caching Strategy dengan Room
    - Data yang diambil dari API akan disimpan ke dalam Room agar tetap bisa diakses saat offline.
    - Saat aplikasi diluncurkan, data akan dimuat dari Room terlebih dahulu dan kemudian disegarkan dari API jika koneksi tersedia.

TAUTAN GIT

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