LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Muhammad Daffa Musyafa **NIM. 2310817110007**

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 Akhir Praktikum ini dikerjakan oleh:

Nama Praktikan : Muhammad Daffa Musyafa
NIM : 2310817110007

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar	Ir. Eka Setya Wijaya, S.T., M.Kom.
NIM. 2210817210012	NIP. 198205082008011010

DAFTAR ISI

DAFTAR GAMBAR

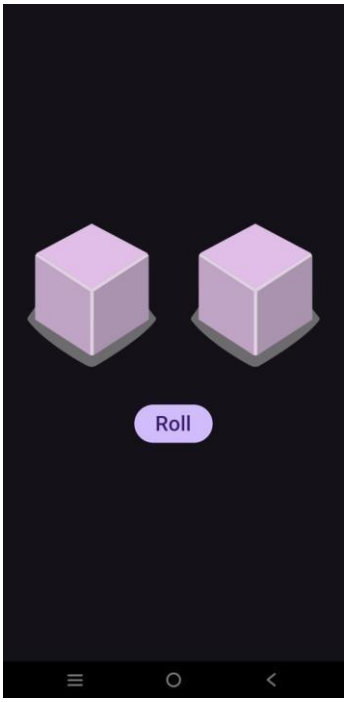
DAFTAR TABEL

MODUL 1 : ANDROID BASIC WITH KOTLIN

SOAL 1

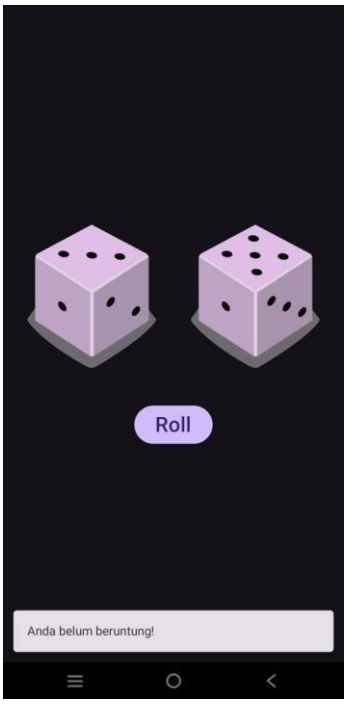
Soal Praktikum:
Buatlah sebuah aplikasi yang dapat menampilkan 2 buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol "Roll". Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1.



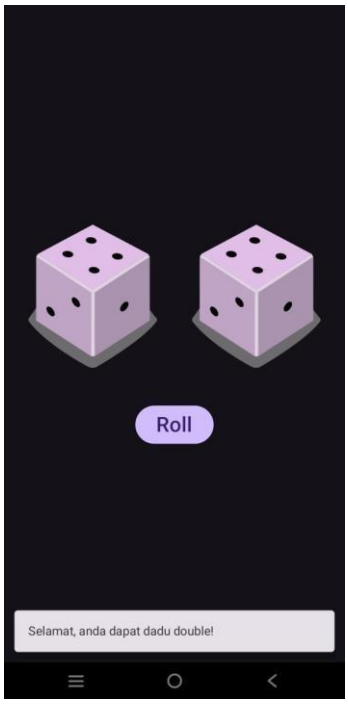
Gambar 1. Tampilan Awal Aplikasi

2. Setelah user menekan tombol "Roll" maka masing-masing dadu akan memperlihatkan sisi dadunya dengan angka antara 1 s/d 6. Apabila user mendapatkan nilai dadu yang berbeda antara Dadu 1 dengan Dadu 2, maka aplikasi akan menampilkan pesan "Anda belum beruntung!" seperti yang dapat dilihat pada Gambar 2.



Gambar 2 Tampilan Dadu Setelah Di-Roll

3. Apabila user mendapatkan nilai dadu yang sama antara Dadu 1 dan Dadu 2 atau nilai double, maka aplikasi akan menampilkan pesan “Selamat, anda dapat dadu double!” seperti yang dapat dilihat pada Gambar 3.



Gambar 3. Tampilan Roll Dadu Double

4. Buatlah aplikasi tersebut menggunakan XML dan Jetpack Compose.
5. Upload aplikasi yang telah anda buat ke dalam repository GitHub ke dalam **folder Modul 1 dalam bentuk Project.** Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repository.

6. Untuk gambar dadu dapat didownload pada link berikut: https://drive.google.com/file/d/14V3qXGdFnuoYN4AGd_9SgFh8kw8X9ySm/view?usp= sharing

A. Source Code XML

MainActivity.kt

Table 1 Source Code Jawaban Soal 1 XML

```
1  package com.example.diceroller
2
```

```
 3 import android.os.Bundle
 4 import android.widget.Button
 5 import android.widget.ImageView
 6 import android.widget.Toast
 7 import androidx.activity.enableEdgeToEdge
 8 import androidx.appcompat.app.AppCompatActivity
 9
10 class MainActivity : AppCompatActivity() {
11     private lateinit var rollbtn: Button
12     private lateinit var dice: ImageView
13     private lateinit var dice2: ImageView
14     private lateinit var teks: String
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         enableEdgeToEdge()
18         setContentView(R.layout.activity_main)
19
20         dice = findViewById(R.id.dice_image1)
21         dice2 = findViewById(R.id.dice_image2)
22
23         rollbtn = findViewById(R.id.roll_button)
24
25
26         rollbtn.setOnClickListener {
27             rollDice()
28         }
29     }
30     private fun rollDice() {
31
32         val randomInt1 = (1..6).random()
```

```
33          val drawableResource1 = when (randomInt1) {
34              1 -> R.drawable.dice_1
35              2 -> R.drawable.dice_2
36              3 -> R.drawable.dice_3
37              4 -> R.drawable.dice_4
38              5 -> R.drawable.dice_5
39              6 -> R.drawable.dice_6
40              else -> R.drawable.dice_0
41          }
42          val randomInt2 = (1..6).random()
43          val drawableResource2 = when (randomInt2) {
44              1 -> R.drawable.dice_1
45              2 -> R.drawable.dice_2
46              3 -> R.drawable.dice_3
47              4 -> R.drawable.dice_4
48              5 -> R.drawable.dice_5
49              6 -> R.drawable.dice_6
50              else -> R.drawable.dice_0
51          }
52          dice.setImageResource(drawableResource1)
53          dice2.setImageResource(drawableResource2)
54
55          if (randomInt1 == randomInt2) {
56              Toast.makeText(this, "Selamat Kamu dapat
57  Double!", Toast.LENGTH_SHORT).show()
58          } else {
59              Toast.makeText(this, "Anda Kurang Beruntung",
60  Toast.LENGTH_SHORT).show()
61          }
62      }}
```

activity_main.xml

Table 2. Source Code Jawaban Soal 1 XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/dice_image1"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginStart="25dp"
        android:src="@drawable/dice_0"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

    <ImageView
        android:id="@+id/dice_image2"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginEnd="25dp"
```

```
        android:src="@drawable/dice_0"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

    <Button
        android:id="@+id/roll_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="250dp"
        android:text="@string/roll"
        android:textSize="20sp"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

B. Output Program

Gambar 4 Screenshot Hasil Jawaban Soal 1 XML

Gambar 5. Screenshot Hasil Jawaban Soal 1 XML

C. Pembahasan

MainActivity.kt:

Pada baris pertama, dideklarasikan nama *package* file Kotlin. Kemudian pada baris 3 hingga 8, dilakukan proses *import* terhadap komponen-komponen UI yang dibutuhkan untuk didefinisikan dalam XML.

Selanjutnya, pada baris 10 hingga 27, terdapat `class MainActivity` yang merupakan turunan dari `AppCompatActivity`, yaitu kelas bawaan Android yang memberikan dukungan untuk fitur-fitur yang kompatibel dengan banyak versi Android. Di dalam kelas ini, dideklarasikan beberapa variabel: `private lateinit var rollbtn: Button` untuk tombol dadu, dan `private lateinit var dice: ImageView` serta `private lateinit var dice2: ImageView` untuk gambar dadu. Keyword `lateinit` menandakan bahwa variabel-variabel ini akan diinisialisasi nanti, biasanya di dalam metode `onCreate`.

Metode `override fun onCreate(savedInstanceState: Bundle?)` merupakan fungsi *lifecycle* yang dipanggil saat activity pertama kali dibuat. Di dalamnya, `enableEdgeToEdge()` digunakan untuk mengaktifkan tampilan layar penuh, dan `setContentView(R.layout.activity_main)` digunakan untuk mengatur layout berdasarkan file `activity_main.xml`. Selanjutnya, variabel `dice` dan `dice2` dihubungkan dengan elemen ImageView di XML menggunakan `findViewById(R.id.dice_image1)` dan `findViewById(R.id.dice_image2)`, di mana fungsi `findViewById()` bertugas mencari komponen berdasarkan ID-nya. Demikian pula, variabel `rollbtn` dihubungkan ke tombol roll melalui ID `roll_button` di XML menggunakan metode yang sama. Setelah itu, ditambahkan `setOnClickListener` pada `rollbtn` untuk menangani aksi saat tombol ditekan, yang akan menjalankan fungsi `rollDice()`.

Pada baris 28 hingga 58, didefinisikan fungsi `private fun rollDice()` yang berfungsi untuk melakukan aksi roll pada dadu. Pertama, dibuat variabel `randomInt1 = (1..6).random()` untuk menghasilkan angka acak antara 1 sampai 6. Kemudian, digunakan struktur `when` untuk menentukan gambar dadu yang sesuai berdasarkan angka acak tersebut dan menyimpannya dalam variabel `drawableResource1`. Gambar dadu diambil dari folder `drawable` seperti `R.drawable.dice_1`, `dice_2`, dan seterusnya hingga `dice_6`, dengan `else` sebagai kondisi default untuk menampilkan gambar dadu kosong (`dice_0`). Hal yang sama dilakukan untuk dadu kedua dengan `val drawableResource2 = when (randomInt2)`.

Setelah gambar dadu ditentukan, masing-masing gambar ditampilkan di layar melalui `dice.setImageResource(drawableResource1)` dan `dice2.setImageResource(drawableResource2)`. Kemudian dibuat kondisi menggunakan `if`, di mana jika `randomInt1 == randomInt2`, maka akan ditampilkan pesan *toast* bertuliskan "Selamat Kamu dapat Double!". Jika tidak sama, maka toast yang ditampilkan bertuliskan "Anda Kurang Beruntung". Toast merupakan notifikasi kecil yang muncul di bagian bawah layar untuk menampilkan pesan sementara.

activity_main.xml:

Untuk bagian `activity_main.xml`, baris pertama berisi standar deklarasi file XML: `<?xml version="1.0" encoding="utf-8"?>`. Pada baris 2 hingga 8 digunakan `ConstraintLayout` yang merupakan layout fleksibel untuk menempatkan elemen-elemen UI berdasarkan posisi elemen lain. Atribut `android:layout_width="match_parent"` dan `android:layout_height="match_parent"` digunakan agar layout

memenuhi seluruh layar. Atribut `tools:context=".MainActivity"` menunjukkan bahwa file XML ini digunakan oleh kelas `MainActivity`.

Pada baris 10 hingga 19, dibuat sebuah `ImageView` untuk menampilkan gambar dadu pertama. Diberikan ID `@+id/dice_image1` dengan ukuran `200dp x 200dp`, serta margin dari kiri sebesar `25dp`. Atribut `android:src="@drawable/dice_0"` digunakan untuk menampilkan gambar dadu kosong sebagai tampilan awal. Posisi gambar diatur menggunakan constraint dari atas, kiri, dan bawah dengan `app:layout_constraintTop_toTopOf="parent"`, `Start_toStartOf`, dan `Bottom_toBottomOf`.

Pada baris 21 hingga 30, dibuat `ImageView` kedua untuk menampilkan gambar dadu kedua. Konsepnya sama dengan `ImageView` pertama, hanya saja constraint-nya berbeda, yaitu menggunakan `layout_constraintEnd_toEndOf="parent"` dan diberi margin dari kanan sebesar `25dp`.

Kemudian pada baris 32 hingga 43 dibuat sebuah tombol menggunakan elemen `Button`. ID dari tombol ini adalah `@+id/roll_button`, dan ukurannya disesuaikan dengan kontennya menggunakan `wrap_content`. Jarak dari atas diberi `layout_marginTop="250dp"` agar tombol berada di bawah gambar dadu. Tulisan pada tombol diambil dari file `strings.xml` melalui atribut `android:text="@string/roll"`, dengan ukuran teks `20sp`. Posisi tombol diatur agar berada di tengah halaman menggunakan constraint atas, bawah, kiri, dan kanan (`Top_toTopOf`, `Bottom_toBottomOf`, `Start_toStartOf`, dan `End_toEndOf`).

Penutup dari layout `ConstraintLayout` ditulis dengan `</androidx.constraintlayout.widget.ConstraintLayout>`.

D. Source Code Compose

MainActivity.kt

Table 3. Source Code Jawaban soal 1 Compose

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

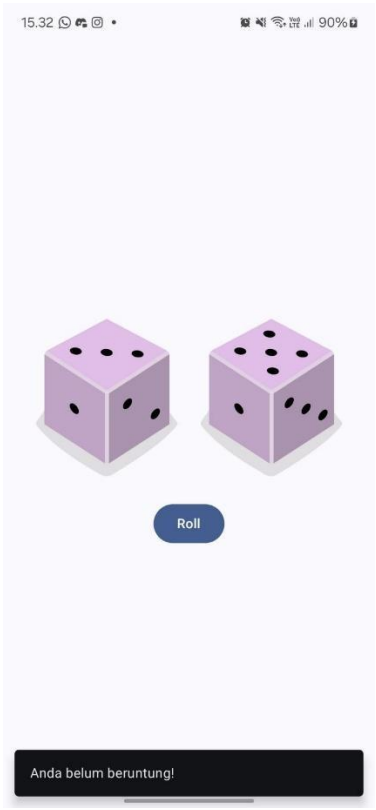
```
 9 import androidx.compose.foundation.layout.Column
10 import androidx.compose.foundation.layout.Row
11 import androidx.compose.foundation.layout.Spacer
12 import androidx.compose.foundation.layout.fillMaxSize
13 import androidx.compose.foundation.layout.padding
14 import androidx.compose.foundation.layout.width
15 import
16 androidx.compose.foundation.layout.wrapContentSize
17 import androidx.compose.material3.Button
18 import androidx.compose.material3.Scaffold
19 import androidx.compose.material3.SnackbarDuration
20 import androidx.compose.material3.SnackbarHost
21 import androidx.compose.material3.SnackbarHostState
22 import androidx.compose.material3.Text
23 import androidx.compose.runtime.Composable
24 import androidx.compose.runtime.getValue
25 import androidx.compose.runtime.mutableStateOf
26 import androidx.compose.runtime.remember
27 import androidx.compose.runtime.rememberCoroutineScope
28 import androidx.compose.runtime.setValue
29 import androidx.compose.ui.Alignment
30 import androidx.compose.ui.Modifier
31 import androidx.compose.ui.res.painterResource
32 import androidx.compose.ui.res.stringResource
33 import androidx.compose.ui.tooling.preview.Preview
34 import androidx.compose.ui.unit.dp
35 import com.example.diceroller.ui.theme.DiceRollerTheme
36 import kotlinx.coroutines.launch
37
38 class MainActivity : ComponentActivity() {
39     override fun onCreate(savedInstanceState: Bundle?) {
40         super.onCreate(savedInstanceState)
41         enableEdgeToEdge()
42         setContent {
43             DiceRollerTheme {
44                 DiceRollerApp()
45             }
46         }
47     }
48
49     @Composable
50     fun DiceWithButtonAndImage(
51         modifier: Modifier,
52         showSnackbar: (String) -> Unit
53     ) {
```

```
54          var result1 by remember { mutableStateOf(0) }
55          val imageResource1 = when (result1) {
56              1 -> R.drawable.dice_1
57              2 -> R.drawable.dice_2
58              3 -> R.drawable.dice_3
59              4 -> R.drawable.dice_4
60              5 -> R.drawable.dice_5
61              6 -> R.drawable.dice_6
62              else -> R.drawable.dice_0
63          }
64          var result2 by remember { mutableStateOf(0) }
65          val imageResource2 = when (result2) {
66              1 -> R.drawable.dice_1
67              2 -> R.drawable.dice_2
68              3 -> R.drawable.dice_3
69              4 -> R.drawable.dice_4
70              5 -> R.drawable.dice_5
71              6 -> R.drawable.dice_6
72              else -> R.drawable.dice_0
73          }
74          Column(
75              modifier = modifier,
76              horizontalAlignment =
77  Alignment.CenterHorizontally
78          ) {
79              Row(
80                  horizontalArrangement =
81  Arrangement.Center,
82                  modifier = Modifier
83              ) {
84                  Image(
85                      modifier = Modifier.width(170.dp),
86                      painter =
87  painterResource(imageResource1),
88                      contentDescription =
89  result1.toString()
90                  )
91                  Image(
92                      modifier = Modifier.width(170.dp),
93                      painter =
94  painterResource(imageResource2),
95                      contentDescription =
96  result2.toString()
97                  )
98              }
```

```
 99
100             Button(onClick = {
101                 result1 = (1..6).random()
102                 result2 = (1..6).random()
103
104                 val message = if (result1 == result2) {
105                     "Selamat anda dapat dadu double!"
106                 } else {
107                     "Anda belum beruntung!"
108                 }
109
110                 showSnackbar(message)
111             }) {
112                 Text("Roll")
113             }
114         }
115
116     }
117
118     @Preview(showBackground = true)
119     @Composable
120     fun DiceRollerApp() {
121         val snackbarHostState = remember {
122 SnackbarHostState() }
123         val corutineScope = rememberCoroutineScope()
124         Scaffold(
125             snackbarHost = {
126 SnackbarHost(snackbarHostState) }
127         ) { paddingValues ->
128             DiceWithButtonAndImage(
129                 modifier = Modifier
130                     .fillMaxSize()
131                     .wrapContentSize(Alignment.Center)
132                     .padding(paddingValues),
133                 showSnackbar = { message ->
134                     corutineScope.launch {
135                         snackbarHostState.showSnackbar(
136                             message = message,
137                             duration =
138 SnackbarDuration.Short
139                         )
140                     }
141                 }
142             )
143         }
```

```
144         }
145 }
```

E. Output Program



Gambar 6. Screenshot Hasil Jawaban Soal 1 Compose

F. Pembahasan

MainActivity.kt:

Pada line 1, dideklarasikan nama package file Kotlin Pada line 3-35, Mengimport komponen komponen UI yang dibutuhkan untuk mendefinisikan dalam compose.

Pada line 37-46, ini main activity, `class MainActivity : ComponentActivity()`, untuk ComponentActivity adalah kelas dari jetpack compose untuk tampilan UI dari kotlin langsung. Kemudian `override fun onCreate(savedInstanceState: Bundle?)`, fungsi *oncreate* adalah

fungsi awal yang dijalankan saat activity, *override* berarti kamu menggantikan fungsi bawaan dari *ComponenActivity* untuk bisa dicostum sendiri. *enableEdgeToEdge()* untuk ini membuat UI *fullscreen*, *setContent {* untuk fungsi UI kamu akan di dalam blok ini, kemudian *DiceRollerTheme {* untuk membuat fungsi tema sendiri berguna untuk memberikan warna, font, bentuk dan lain lain, dengan isi *DiceRollerApp()* untuk memuat *Composable function utama.* Pada line 48-109, `@Composable` ini untuk fungsi UI tampilan di jetpack compose.

Kemudian `fun DiceWithButtonAndImage( modifier: Modifier,showSnackbar: (String) -> Unit )` membuat fungsi *DiceWithButtonAndImage* dengan alat bantu buat atur ukuran, posisi, padding, dan lain-lain menggunkan *modifier,* kemudian `showSnackbar` adalah lambda function untuk menampilkan Snackbar, di panggil setelah tombol ditekan. Kemudian membuat variabel var `result1 by remember { mutableStateOf(0) }` dan `var result2 by remember { mutableStateOf(0) }` untuk menyimpan angka acak untuk dadu 1 dan dadu 2, untuk remember + mutableStatof() nilai ynag bisa berubah, dan akan rekomposisi UI kalau nilainya berubah. Kemudian membuat variabel `val imageResource1 = when (result1){ ... }` dan `val imageResource2 = when (result2) { ... }` memilih gambar dadu berdasarkan angka yang muncul di result1 sama 1 sampai 6 dengan gambar masing masing `1 -> R.drawable.dice_1`, jika keluar 1 maka maenampilkan dadu 1.

Kemudian pada bagian layout membaut `column` menyusun elemen secara vertikal: gambar + tombol, kemudian membuat columnya rata tengah horizontal dengan `horizontalAlignment = Alignment.CenterHorizontally`. Kemudian membuat row utnuk gambar kedua dengan Row(...) { ... }, Di dalam *Column*, kamu buat *Row* untuk meletakkan 2 dadu secara horizontal dengan posisi dadu dibuat berada di tengah baris.

Kemudian membuat `Image(…)` dibuat 2 untuk menampilkan dadu 1 dan dadu 2, dengan mengambil gambar menggunakan `painterResource(...)`, mengambil dari drawable. Kemudian `modifier = Modifier.width(170.dp)` untuk mengatur lebar gambar agar tidak terlalu over, kemudian `contentDescription = result1.toString()` untuk buat aksesibilitas. Kemudian membuat button dengan `Button(onClick = { ... })`, jadi ketika tombol di klik akan mengacak angka dadu dengan `result1 = (1..6).random()` dan `result2 = (1..6).random()`, kemudian membuat kondisi dengan if `val message = if (result1 == result2)`, jika result1 sama dengan result2 maka akan mengeluarkan "Selamat anda dapat dadu double!" jika beda "Anda belum beruntung!". Kemudian menampikan *snackbar* dengan `showSnackbar(message)`. Kemudian membuat text pada *button* dengan `Text("Roll")` dengan isi Roll.

Pada line 111-135, `@Preview(showBackground = true)` untuk membuat compose bisa ditampilkan di android studio, kemudian `@composable` untuk UI compose. Kemudian membuat variabel `val snackbarHostState = remember {SnackbarHostState()}` dan `val corutineScope = rememberCoroutineScope()` untuk mengontrol snackbar, kapan ditampilkan dan isinya apa, kemudian remember {…} menyimpan state agar tidak reset saat *recomposition*. Kemudian `rememberCoroutineScope()` untuk bikin scope corutine supaya bisa menampilkan *snackbar* secara *asynchronous*. Kemudian membuat `Scaffold( snackbarHost ={SnackbarHost(snackbarHostState) }`, ini adalah layout Compose lengkap, kemudain `snackbarHost = {SnackbarHost(snackbarHostState)` untuk mengubungkan*snackbarhoststate* supaya compose tahu di mana menampilkan *snackbar.* Kemudian membuat isi dari Scaffold dengan `paddingValues` Otomatis diberikan oleh *Scaffold* untuk menghindari ketabrak status bar navigation.

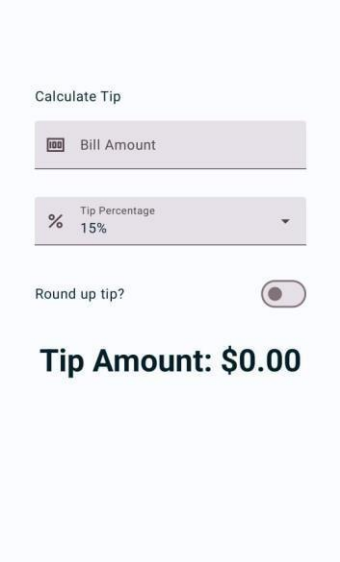
Kemudian isi dari `DiceWithButtonAndImage(` `Modifier.fillMaxSize()` untuk ukuran yang seluar layar, kemudain `wrapContentSize(Alignment.Center)` posisi konten di tengah layar, kemudian . `padding(paddingValues)` untuk kasih padding bawaan scaffold. Kemudian `showSnackbar` untuk menampilkan isi dari snackbar dan durasinya pendek dengan `SnackbarDuration.Short`.

MODUL 2: ANDROID LAYOUT WITH COMPOSE

SOAL 1

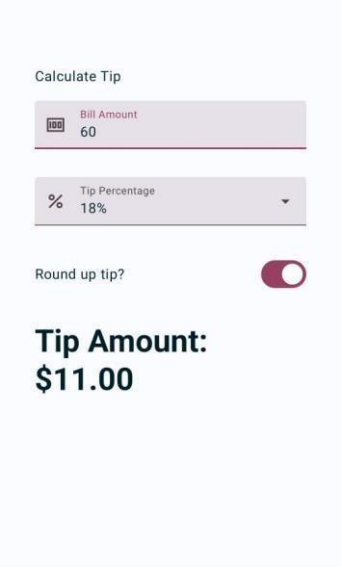
Soal Praktikum:
Buatlah sebuah aplikasi kalkulator tip menggunakan XML dan Jetpack Compose yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

A. Input biaya layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
B. Pilihan persentase tip: Pengguna dapat memilih persentase tip yang diinginkan.
C. Pengaturan pembulatan tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
D. Tampilan hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.



Gambar 7 Tampilan Awal Aplikasi

Gambar 8 Tampilan Pilihan Persentase Tip



Gambar 9 Tampilan Pilihan Persentase Tip

A. **Source Code XML**

Table 4 Source Code Jawaban soal 1 XML

```
 1 package com.example.calculatortipx
 2
 3 import android.os.Bundle
 4 import android.text.Editable
 5 import android.text.TextWatcher
 6 import android.view.View
 7 import android.view.ViewGroup
 8 import android.widget.AdapterView
 9 import android.widget.ArrayAdapter
10 import android.widget.TextView
11 import androidx.appcompat.app.AppCompatActivity
12 import
13 com.example.calculatortipx.databinding.ActivityMainBinding
14 import java.text.NumberFormat
15 import java.util.*
16 import kotlin.math.ceil
17
18
19 class MainActivity : AppCompatActivity() {
20
21     private lateinit var binding: ActivityMainBinding
22
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         binding = ActivityMainBinding.inflate(layoutInflater)
26         setContentView(binding.root)
27
28         val tipOptions =
29 resources.getStringArray(R.array.tip_options)
30
31         val adapter = object : ArrayAdapter<String>(
32             this,
33             R.layout.spinner_item_with_icon,
```

```
            R.id.text1,
            tipOptions
        ) {
            override fun getView(position: Int, convertView:
View?, parent: ViewGroup): View {
                val view =
layoutInflater.inflate(R.layout.spinner_item_with_icon, parent,
false)
                val label =
view.findViewById<TextView>(R.id.label)
                val text =
view.findViewById<TextView>(R.id.text1)
                label.text = "Tip Percentage"
                text.text = tipOptions[position]
                return view
            }

            override fun getDropDownView(position: Int,
convertView: View?, parent: ViewGroup): View {
                val view =
layoutInflater.inflate(R.layout.spinner_dropdown_item_with_icon,
parent, false)
                val text =
view.findViewById<TextView>(R.id.text1)
                text.text = tipOptions[position]
                return view
            }
        }
        binding.tipOptionsSpinner.adapter = adapter
        binding.tipOptionsSpinner.setSelection(0)
        calculateTip() // panggil saat awal agar 15% dihitung

binding.serviceCostInput.addTextChangedListener(tipWatcher)
```

```
        binding.tipOptionsSpinner.onItemSelectedListener =
spinnerListener
        binding.roundUpSwitch.setOnCheckedChangeListener { _, _
-> calculateTip() }
    }

    private val tipWatcher = object : TextWatcher {
        override fun afterTextChanged(s: Editable?) =
calculateTip()
        override fun beforeTextChanged(s: CharSequence?, start:
Int, count: Int, after: Int) {}
        override fun onTextChanged(s: CharSequence?, start: Int,
before: Int, count: Int) {}
    }

    private val spinnerListener = object :
AdapterView.OnItemSelectedListener {
        override fun onItemSelected(parent: AdapterView<*>?,
view: View?, position: Int, id: Long) {
            calculateTip()
        }
        override fun onNothingSelected(parent: AdapterView<*>?)
{}
    }

    private fun calculateTip() {
        val cost =
binding.serviceCostInput.text.toString().toDoubleOrNull() ?: 0.0

        val tipPercent =
binding.tipOptionsSpinner.selectedItem.toString().removeSuffix("
%").toDoubleOrNull() ?: 0.0
        var tip = cost * tipPercent / 100
```

```
99          if (binding.roundUpSwitch.isChecked) {
100             tip = ceil(tip)
101         }
102         val formattedTip =
NumberFormat.getCurrencyInstance(Locale.US).format(tip)
103         binding.tipResult.text = "Tip Amount: $formattedTip"
104     }
105 }
106
107
108
```

activity_main.xml :

Table 5 Source Code Jawaban soal 1 XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3
4  xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:padding="24dp"
9      android:background="@color/background_light">
10
11     <TextView
12         android:id="@+id/titleText"
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:text="Calculate Tip"
16         android:textSize="20sp"
```

```
        android:textColor="@color/primary_text"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <EditText
        android:id="@+id/serviceCostInput"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Bill Amount"
        android:inputType="numberDecimal"
        android:drawableStart="@drawable/money"
        android:drawablePadding="8dp"
        android:background="@drawable/edittext_background"
        android:padding="12dp"
        android:textColor="@color/primary_text"
        android:textSize="16sp"
        app:layout_constraintTop_toBottomOf="@id/titleText"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp"/>

    <Spinner
        android:id="@+id/tipOptionsSpinner"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:background="@drawable/edittext_background"
        android:minHeight="48dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
47
48  app:layout_constraintTop_toBottomOf="@id/serviceCostInput"
49  />
50
51      <TextView
52          android:id="@+id/roundUpLabel"
53          android:layout_width="wrap_content"
54          android:layout_height="wrap_content"
55          android:text="Round up tip?"
56
57  app:layout_constraintTop_toBottomOf="@id/tipOptionsSpinner"
58          app:layout_constraintStart_toStartOf="parent"
59          android:textSize="20dp"
60          android:layout_marginTop="16dp" />
61
62      <Switch
63          android:id="@+id/roundUpSwitch"
64          android:layout_width="wrap_content"
65          android:layout_height="wrap_content"
66          android:layout_marginTop="16dp"
67          android:minWidth="48dp"
68          android:minHeight="48dp"
69          app:layout_constraintEnd_toEndOf="parent"
70
71  app:layout_constraintTop_toBottomOf="@id/tipOptionsSpinner"
72  />
73
74      <TextView
75          android:id="@+id/tipResult"
76          android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="Tip Amount: $0.00"
        android:textStyle="bold"
        android:textSize="22sp"
        android:textColor="@color/tip_result_text"
        android:layout_marginTop="32dp"

app:layout_constraintTop_toBottomOf="@id/roundUpSwitch"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

spinner_item_with_icon.xml:

Table 6 Source Code Jawaban soal 1 XML

```
package com.example.calculatortipx

import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.view.View
import android.view.ViewGroup
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import
com.example.calculatortipx.databinding.ActivityMainBinding
import java.text.NumberFormat
import java.util.*
```

```
import kotlin.math.ceil


class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val tipOptions =
resources.getStringArray(R.array.tip_options)

        val adapter = object : ArrayAdapter<String>(
            this,
            R.layout.spinner_item_with_icon,
            R.id.text1,
            tipOptions
        ) {
            override fun getView(position: Int, convertView:
View?, parent: ViewGroup): View {
                val view =
layoutInflater.inflate(R.layout.spinner_item_with_icon, parent,
false)
                val label =
view.findViewById<TextView>(R.id.label)
                val text =
view.findViewById<TextView>(R.id.text1)
                label.text = "Tip Percentage"
                text.text = tipOptions[position]
                return view
```

```
            }

            override fun getDropDownView(position: Int,
convertView: View?, parent: ViewGroup): View {
                val view =
layoutInflater.inflate(R.layout.spinner_dropdown_item_with_icon,
parent, false)
                val text =
view.findViewById<TextView>(R.id.text1)
                text.text = tipOptions[position]
                return view
            }
        }
        binding.tipOptionsSpinner.adapter = adapter
        binding.tipOptionsSpinner.setSelection(0)
        calculateTip() // panggil saat awal agar 15% dihitung

binding.serviceCostInput.addTextChangedListener(tipWatcher)
        binding.tipOptionsSpinner.onItemSelectedListener =
spinnerListener
        binding.roundUpSwitch.setOnCheckedChangeListener { _, _
-> calculateTip() }
    }

    private val tipWatcher = object : TextWatcher {
        override fun afterTextChanged(s: Editable?) =
calculateTip()
        override fun beforeTextChanged(s: CharSequence?, start:
Int, count: Int, after: Int) {}
        override fun onTextChanged(s: CharSequence?, start: Int,
before: Int, count: Int) {}
    }

```

```
    private val spinnerListener = object :
AdapterView.OnItemSelectedListener {
        override fun onItemSelected(parent: AdapterView<*>?,
view: View?, position: Int, id: Long) {
            calculateTip()
        }
        override fun onNothingSelected(parent: AdapterView<*>?)
{}
    }

    private fun calculateTip() {
        val cost =
binding.serviceCostInput.text.toString().toDoubleOrNull() ?: 0.0

        val tipPercent =
binding.tipOptionsSpinner.selectedItem.toString().removeSuffix("
%").toDoubleOrNull() ?: 0.0
        var tip = cost * tipPercent / 100
        if (binding.roundUpSwitch.isChecked) {
            tip = ceil(tip)
        }
        val formattedTip =
NumberFormat.getCurrencyInstance(Locale.US).format(tip)
        binding.tipResult.text = "Tip Amount: $formattedTip"
    }
}
```

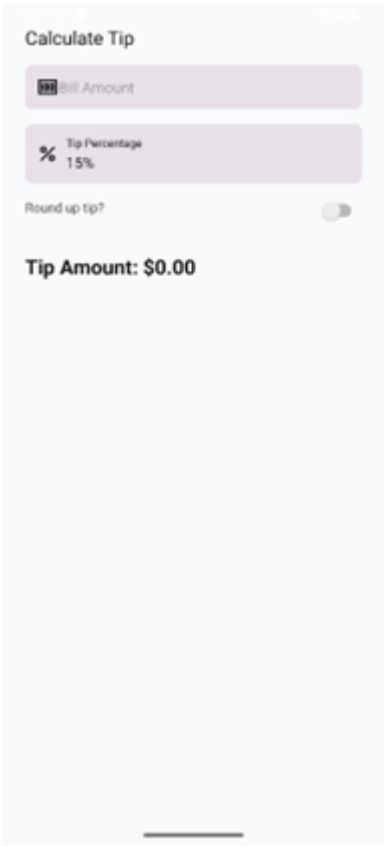
spinner_dropdown_item_with_icon.xml:

Table 7 Source Code Jawaban soal 1 XML

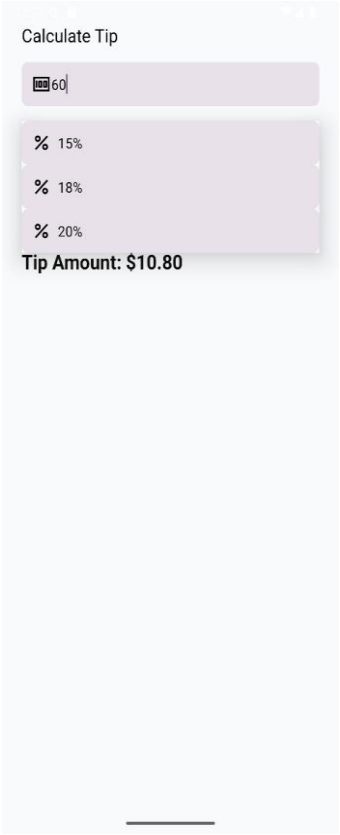
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="12dp"
    android:background="@drawable/edittext_background"
    android:gravity="center_vertical">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="24dp"
        android:src="@drawable/percent"
        android:layout_marginEnd="8dp" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Tip Percentage"
            android:textColor="@color/primary_text"
            android:textSize="12sp" />

        <TextView
            android:id="@+id/text1"
            android:layout_width="wrap_content"
```

```
32              android:layout_height="wrap_content"
33              android:textColor="@color/primary_text"
34              android:textSize="16sp"
35              android:paddingTop="2dp" />
36      </LinearLayout>
37  </LinearLayout>
```

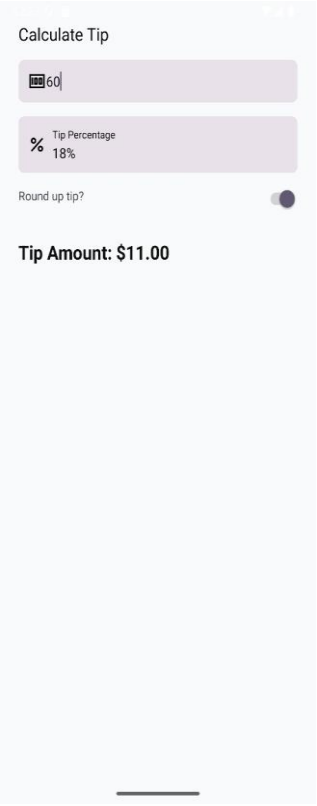
B. **Output Program**



Gambar 10 Screenshot Hasil Jawaban Soal 1 XML



Gambar 11 Screenshot Hasil Jawaban Soal 1 XML



Gambar 12 Screenshot Hasil Jawaban Soal 1 XML

C. Pembahasan

MainActivity.kt:

Pada line 16–70, didefinisikan class MainActivity yang merupakan turunan dari AppCompatActivity, yaitu activity utama dalam aplikasi ini. Di dalamnya terdapat property binding yang digunakan untuk mengakses elemen-elemen UI dari layout *activity_main.xml*.

Pada fungsi `onCreate`, pertama-tama binding diinisialisasi dengan `ActivityMainBinding.inflate` dan ditampilkan ke layar menggunakan `setContentView(binding.root)`. Kemudian, array `tipOptions` yang berisi daftar persentase tip `(seperti 15%, 18%, 20%)` diambil dari resource *strings.xml*. Setelah itu dibuatlah adapter kustom dari ArrayAdapter, menggunakan layout *spinner_item_with_icon* sebagai tampilan item spinner. Fungsi `getView` di

dalam adapter digunakan untuk menampilkan layout spinner ketika tidak diklik, dengan label "Tip Percentage" dan teks persen yang diambil dari `tipOptions`. Sedangkan fungsi getDropDownView digunakan untuk menampilkan item *dropdown* ketika spinner dibuka, hanya menampilkan teks persen. Setelah adapter selesai dikonfigurasi, spinner `tipOptionsSpinner` dihubungkan dengan adapter, dan dipilih default ke indeks 0 agar langsung menampilkan nilai *15%* pada awal aplikasi dijalankan. Fungsi `calculateTip()` langsung dipanggil untuk menghitung tip awal berdasarkan nilai default. Selanjutnya, listener ditambahkan ke `EditText serviceCostInput` agar perhitungan tip bisa dilakukan setiap kali pengguna mengubah nominal biaya layanan. Spinner juga diberi listener untuk mendeteksi perubahan pilihan persentase tip. Terakhir, *switch* `roundUpSwitch` diberikan listener untuk menghitung ulang tip ketika pengguna mengaktifkan atau menonaktifkan pembulatan.

Pada line 47–52, didefinisikan objek `tipWatcher` sebagai implementasi `TextWatcher`. Objek ini digunakan untuk mendeteksi perubahan teks pada input biaya layanan. Hanya metode `afterTextChanged` yang diisi, yang akan memanggil fungsi `calculateTip()` setiap kali pengguna selesai mengetik.

Pada line 54–58, didefinisikan objek `spinnerListener` yang merupakan implementasi dari `AdapterView.OnItemSelectedListener`. Listener ini digunakan untuk menghitung ulang tip saat pengguna memilih persentase baru dari spinner. Fungsi `calculateTip()` dipanggil dalam onItemSelected, sedangkan fungsi onNothingSelected tidak melakukan apapun.

Pada line 60–70, didefinisikan fungsi `calculateTip()` sebagai inti logika perhitungan tip. Pertama, nominal biaya layanan dibaca dari `EditText` dan dikonversi ke Double. Jika input kosong atau tidak valid, maka akan dianggap sebagai 0. Kemudian persentase tip diambil dari item yang dipilih di spinner, dengan menghapus simbol persen *"%"* dan dikonversi menjadi nilai desimal. Nilai tip

dihitung dengan mengalikan biaya dengan persen tip dan akan dibulatkan ke atas menggunakan fungsi ceil apabila switch `roundUpSwitch` diaktifkan. Hasil akhir dari tip diformat menggunakan `NumberFormat` dengan locale Amerika Serikat dan ditampilkan dalam bentuk teks pada `TextView` dengan format `Tip Amount: $[nominal]`.

activity_main.xml:

Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8. Selanjutnya pada line 2–5, digunakan tag `ConstraintLayout` dari Jetpack ConstraintLayout sebagai layout utama dengan namespace Android (`xmlns:android`) dan App (`xmlns:app`) yang digunakan untuk mengatur atribut-atribut khusus. Layout ini memiliki ukuran penuh terhadap parent (`match_parent`) baik untuk lebar maupun tinggi, ditambahkan padding sebesar 24dp di seluruh sisi, serta latar belakang diatur menggunakan warna `@color/background_light`.

Pada line 7–14, ditambahkan komponen `TextView` dengan id `titleText`, yang berfungsi sebagai judul tampilan dengan teks "Calculate Tip". Ukuran teks diatur sebesar 20sp dan warnanya menggunakan warna primer `@color/primary_text`. Komponen ini diposisikan di bagian atas dan kiri layout menggunakan constraint `layout_constraintTop_toTopOf="parent"` dan `layout_constraintStart_toStartOf="parent"`.

Pada line 16–28, didefinisikan komponen `EditText` dengan id `serviceCostInput` yang digunakan untuk menginput nominal tagihan atau biaya layanan. Lebar elemen diatur `0dp` agar mengikuti constraint start dan end (disebut match constraints), sementara tinggi mengikuti kontennya. EditText ini

memiliki hint "Bill Amount", tipe input `numberDecimal` agar hanya menerima angka desimal, serta icon di sisi kiri yang berasal dari drawable `money`. Selain itu, terdapat padding internal sebesar 12dp, padding antara icon dan teks sebesar 8dp, latar belakang khusus yang ditentukan oleh `@drawable/edittext_background`, dan warna teks menggunakan `@color/primary_text`. Posisi EditText ini diatur tepat di bawah titleText dengan margin atas sebesar 16dp dan direntangkan dari kiri ke kanan layout.

Pada line 30–38, didefinisikan komponen `Spinner` dengan id `tipOptionsSpinner` yang digunakan sebagai dropdown untuk memilih persentase tip. Lebarnya juga diatur `0dp` agar mengikuti constraint kiri dan kanan, dan tinggi menyesuaikan isi. Spinner ini memiliki margin atas 16dp dari `serviceCostInput`, latar belakang dari drawable kustom `edittext_background`, serta tinggi minimum 48dp untuk menjaga konsistensi ukuran elemen.

Pada line 40–46, ditambahkan `TextView` dengan id `roundUpLabel` yang menampilkan teks "Round up tip?" untuk menanyakan apakah hasil tip perlu dibulatkan. Ukuran teks diatur sebesar 20sp, dan posisinya diatur tepat di bawah Spinner `tipOptionsSpinner`, dengan margin atas 16dp dan disejajarkan ke kiri layout.

Pada line 48–54, didefinisikan komponen `Switch` dengan id `roundUpSwitch` yang digunakan pengguna untuk mengaktifkan atau menonaktifkan fitur pembulatan tip. Switch ini memiliki margin atas 16dp, tinggi dan lebar minimum masing-masing 48dp untuk memastikannya cukup besar untuk disentuh, dan posisinya disejajarkan ke kanan layout, tepat di bawah `tipOptionsSpinner`.

Pada line 56–63, didefinisikan `TextView` dengan id `tipResult` yang digunakan untuk menampilkan hasil akhir perhitungan tip. Teks default-nya adalah "Tip Amount: $0.00", dengan ukuran teks sebesar 22sp dan gaya teks tebal. Warna

teksnya diatur menggunakan `@color/tip_result_text`. TextView ini diposisikan di bawah Switch `roundUpSwitch` dengan margin atas 32dp dan disejajarkan ke kiri layout.

spinner_item_with_icon.xml:

Pada line 1–8, digunakan elemen `LinearLayout` sebagai root layout dengan orientasi horizontal. Layout ini diatur agar memiliki lebar penuh (`match_parent`) dan tinggi menyesuaikan isi (`wrap_content`). Padding sebesar 12dp ditambahkan ke seluruh sisi layout untuk memberi ruang di dalam elemen, dan latar belakangnya menggunakan drawable kustom `@drawable/edittext_background`, yang kemungkinan memberikan border atau efek khusus. Atribut `gravity="center_vertical"` digunakan agar konten di dalam layout disejajarkan secara vertikal di tengah.

Pada line 10–13, ditambahkan komponen `ImageView` yang menampilkan ikon bergambar persentase (`@drawable/percent`) dengan ukuran 24x24dp. Margin end sebesar 8dp ditambahkan agar ada jarak antara gambar dan elemen di sebelah kanannya.

Pada line 15–22, terdapat `LinearLayout` kedua di dalam layout utama, dengan orientasi vertikal. Layout ini digunakan untuk menumpuk dua `TextView` secara vertikal, yaitu label dan nilai persen tip. Lebar dan tinggi layout ini disesuaikan dengan ukuran kontennya (`wrap_content`).

Pada line 24–29, dideklarasikan `TextView` pertama dengan id `label`, yang berfungsi menampilkan label teks "Tip Percentage". Ukuran teks diatur sebesar 12sp dan warna teks menggunakan warna `@color/primary_text`.

Pada line 31–36, didefinisikan `TextView` kedua dengan id `text1` yang berfungsi menampilkan nilai persen tip yang dipilih, seperti "15%", "18%", atau "20%".

Ukuran teksnya sedikit lebih besar, yaitu 16sp, dan warna teksnya sama seperti label. Padding atas sebesar 2dp ditambahkan agar ada jarak antara label dan nilai teks ini.

spinner_dropdown_item_with_icon.xml:

Pada line 1–8, digunakan elemen `LinearLayout` sebagai layout utama dengan orientasi horizontal, yang berarti elemen-elemen di dalamnya akan ditata secara mendatar dari kiri ke kanan. Layout ini memiliki lebar penuh (`match_parent`) dan tinggi yang menyesuaikan isi kontennya (`wrap_content`). Padding sebesar 12dp ditambahkan ke seluruh sisi untuk memberi ruang di dalam layout, dan atribut `gravity="center_vertical"` digunakan agar konten disejajarkan secara vertikal di tengah. Selain itu, latar belakang layout menggunakan drawable `@drawable/edittext_background`, kemungkinan besar untuk memberikan efek visual seperti border atau bayangan agar terlihat seperti input field.

Pada line 10–13, dideklarasikan sebuah `ImageView` yang digunakan untuk menampilkan ikon bergambar persentase (`@drawable/percent`). Ukuran ikon ditetapkan sebesar 24x24dp dan diberi `layout_marginEnd` sebesar 8dp untuk memberikan jarak antara ikon dan elemen di sebelah kanannya, sehingga tata letaknya tetap rapi.

Pada line 15–20, terdapat `TextView` dengan id `text1` yang digunakan untuk menampilkan nilai teks dari opsi persentase tip yang dipilih, seperti "15%", "18%", atau "20%". Ukuran teks ditetapkan sebesar 16sp agar mudah terbaca, dan warna teks ditentukan dengan menggunakan warna `@color/primary_text`, agar selaras dengan tema aplikasi.

A. Source Code Compose

Table 8 Source Code Jawaban soal 1 Compose

```
 1 package com.example.calculatortipj
 2
 3 import android.os.Bundle
 4 import androidx.activity.ComponentActivity
 5 import androidx.activity.compose.setContent
 6 import androidx.activity.enableEdgeToEdge
 7 import androidx.compose.foundation.layout.Arrangement
 8 import androidx.compose.foundation.layout.Column
 9 import androidx.compose.foundation.layout.Spacer
10 import androidx.compose.foundation.layout.fillMaxWidth
11 import androidx.compose.foundation.layout.height
12 import androidx.compose.foundation.layout.padding
13 import androidx.compose.foundation.rememberScrollState
14 import androidx.compose.foundation.text.KeyboardOptions
15 import androidx.compose.foundation.verticalScroll
16 import androidx.compose.material3.MaterialTheme
17 import androidx.compose.material3.Text
18 import androidx.compose.material3.TextField
19 import androidx.compose.runtime.Composable
20 import androidx.compose.runtime.getValue
21 import androidx.compose.runtime.mutableStateOf
22 import androidx.compose.runtime.saveable.rememberSaveable
23 import androidx.compose.runtime.setValue
24 import androidx.compose.ui.Alignment
25 import androidx.compose.ui.Modifier
26 import androidx.compose.ui.res.painterResource
27 import androidx.compose.ui.text.input.KeyboardType
```

```
28 import androidx.compose.ui.tooling.preview.Preview
29 import androidx.compose.ui.unit.dp
30 import com.example.calculatortipj.ui.theme.CalculatorTipJTheme
31 import androidx.compose.material3.Icon
32 import androidx.compose.material3.Switch
33 import androidx.compose.foundation.layout.Row
34 import androidx.compose.ui.text.input.ImeAction
35 import java.text.NumberFormat
36 import androidx.compose.material3.*
37 import androidx.compose.material3.ExposedDropdownMenuBox
38 import androidx.compose.material3.ExposedDropdownMenuDefaults
39 import androidx.compose.runtime.remember
40
41
42 class MainActivity : ComponentActivity() {
43     override fun onCreate(savedInstanceState: Bundle?) {
44         super.onCreate(savedInstanceState)
45         enableEdgeToEdge()
46         setContent {
47             CalculatorTipJTheme {
48                 CalculatorTipApp()
49             }
50         }
51     }
52 }
53
54 @Composable
55 fun CalculatorTipApp() {
56     var amountInput by rememberSaveable { mutableStateOf("") }
57
```

```
    var selectedTipPercent by rememberSaveable { mutableStateOf("15%") }
    var roundUp by rememberSaveable { mutableStateOf(false) }

    val amount = amountInput.toDoubleOrNull() ?: 0.0
    val tipPercent = selectedTipPercent.removeSuffix("%").toDoubleOrNull() ?: 0.0
    val tip = calculateTip(amount, tipPercent, roundUp)

    Column(
        modifier = Modifier
            .verticalScroll(rememberScrollState())
            .padding(40.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            text = "Calculator Tip",
            modifier = Modifier
                .padding(bottom = 16.dp, top = 16.dp)
                .align(alignment = Alignment.Start)
        )
        EditNumberField(
            label = "Bill Amount",
            value = amountInput,
            leadingIcon = R.drawable.money,
            onValueChange = { amountInput = it },
            keyboardOptions = KeyboardOptions(
                keyboardType = KeyboardType.Number,
                imeAction = ImeAction.Next
```

```
            ),
            modifier = Modifier
                .padding(bottom = 32.dp)
                .fillMaxWidth()
        )
        TipPercentageDropdown(
            selectedOption = selectedTipPercent,
            onOptionSelected = { selectedTipPercent = it },
            modifier = Modifier
                .padding(bottom = 32.dp)
                .fillMaxWidth()

        )
        roundTheTipRow(
            roundUp = roundUp,
            onCheckedChange = { roundUp = it },
            modifier = Modifier
                .padding(bottom = 32.dp)
        )
        Text(
            text = "Tip Amount: $tip",
            style = MaterialTheme.typography.displaySmall,
        )
        Spacer(modifier = Modifier.height(150.dp))
    }
}

@Composable
fun EditNumberField(
    label: String,
```

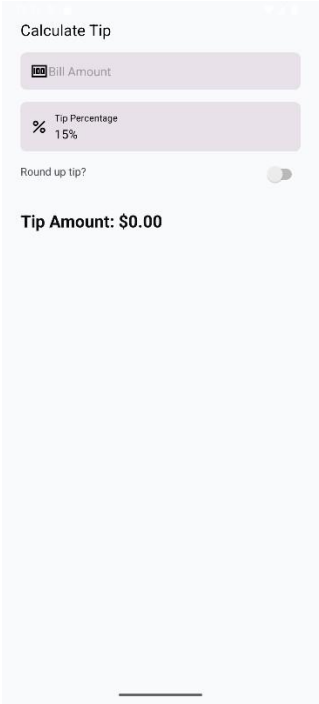
```
118     value: String,
119     leadingIcon: Int,
120     onValueChange: (String) -> Unit,
121     keyboardOptions: KeyboardOptions,
122     modifier: Modifier = Modifier
123 ) {
124     TextField(
125         value = value,
126         singleLine = true,
127         modifier = modifier,
128         onValueChange = onValueChange,
129         label = { Text(text = label) },
130         leadingIcon = { Icon(painterResource(id = leadingIcon),
131 contentDescription = null) },
132         keyboardOptions = keyboardOptions
133     )
134 }
135
136 @Composable
137 fun roundTheTipRow(
138     roundUp: Boolean,
139     onCheckedChange: (Boolean) -> Unit,
140     modifier: Modifier = Modifier
141 ) {
142     Row(
143         modifier = modifier
144             .fillMaxWidth()
145             .padding(16.dp),
146         verticalAlignment = Alignment.CenterVertically,
147         horizontalArrangement = Arrangement.SpaceBetween
```

```
148     ) {
149         Text(text = "Round up tip")
150         Switch(
151             checked = roundUp,
152             onCheckedChange = onCheckedChange
153         )
154     }
155 }
156
157 private fun calculateTip(amount: Double, tipPercent: Double,
158 roundUp: Boolean): String {
159     var tip = tipPercent / 100 * amount
160     if (roundUp) {
161         tip = kotlin.math.ceil(tip)
162     }
163     return
164 NumberFormat.getCurrencyInstance(java.util.Locale.US).format(
165 tip)
166 }
167
168 @Preview(showBackground = true)
169 @Composable
170 fun DefaultPreview() {
171     CalculatorTipJTheme {
172         CalculatorTipApp()
173     }
174 }
175
176 @OptIn(ExperimentalMaterial3Api::class)
177 @Composable
```

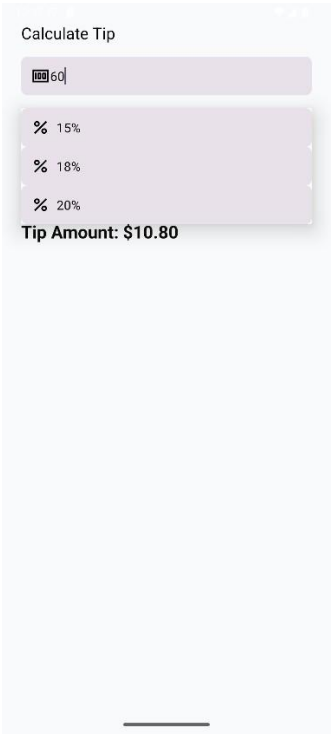
```
178 fun TipPercentageDropdown(
179     selectedOption: String,
180     onOptionSelected: (String) -> Unit,
181     modifier: Modifier = Modifier
182 ) {
183     val options = listOf("15%", "18%", "20%")
184     var expanded by remember { mutableStateOf(false) }
185 
186     ExposedDropdownMenuBox(
187         expanded = expanded,
188         onExpandedChange = { expanded = !expanded },
189         modifier = modifier
190     ) {
191         TextField(
192             value = selectedOption,
193             onValueChange = {},
194             readOnly = true,
195             label = { Text("Tip Percentage") },
196             trailingIcon = {
197 
198 ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded)
199             },
200             modifier = Modifier.menuAnchor().fillMaxWidth()
201         )
202 
203         ExposedDropdownMenu(
204             expanded = expanded,
205             onDismissRequest = { expanded = false }
206         ) {
207             options.forEach { option ->
```

```
208                     DropdownMenuItem(
209                         text = { Text(option) },
210                         onClick = {
211                             onOptionSelected(option)
212                             expanded = false
213                         }
214                     )
215                 }
216             }
217         }
    }
```

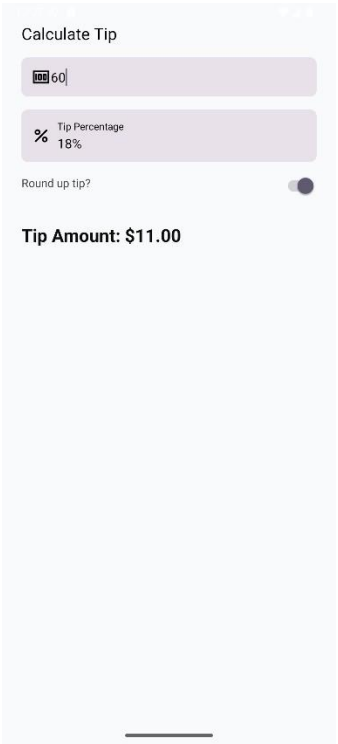
B. Output Program



Gambar 13 Screenshot Hasil Jawaban Soal 1



Gambar 14 Screenshot Hasil Jawaban Soal 1



Gambar 15 Screenshot Hasil Jawaban Soal 1

C. Pembahasan

MainActivity.kt:

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.calculatortipj`. Pada line 3–39, dilakukan import terhadap berbagai komponen dan library JetpackCompose seperti `Column`, `TextField`, `Text`, `Modifier`, dan `MaterialTheme`, serta beberapa komponen tambahan seperti `Switch`, `Icon`, dan `KeyboardOptions` yang dibutuhkan untuk membangun UI aplikasi menggunakan Jetpack Compose, bukan XML.

Pada line 54-112, didefinisikan class `MainActivity` yang merupakan turunan dari `ComponentActivity`, yaitu activity dasar untuk aplikasi yang menggunakan JetpackCompose.Di dalam fungsi `onCreate`, dipanggil fungsi `enableEdgeToEdge()` agar aplikasi tampil full screen. Lalu, `setContent` digunakan untuk mengatur konten UI dari aplikasi, dan fungsi `CalculatorTipApp()` dipanggil sebagai entry point untuk komponen Compose utama. Kemudian didefinisikan fungsi `CalculatorTipApp()` sebagai komponen utama UI aplikasi. Di dalamnya dideklarasikan beberapa state seperti `amountInput`, `selectedTipPercent`, dan `roundUp` dengan `rememberSaveable`, agar state tetap terjaga saat konfigurasi berubah. Kemudian dihitung jumlah tip menggunakan fungsi `calculateTip()`. UI ditampilkan dalam `Column` dengan pengaturan scroll dan padding.

Pada baris 113-133 Komponen yang ditampilkan di dalamnya antara lain: judul aplikasi dengan `Text`, input nominal tagihan dengan `EditNumberField`, dropdown pilihan persen tip dengan `TipPercentageDropdown`, toggle switch pembulatan tip dengan `roundTheTipRow`, serta hasil tip yang dihitung.Kemudain didefinisikan fungsi `EditNumberField()` untuk membuat input angka nominal tagihan. Fungsi ini menampilkan `TextField` dengan icon di bagian depan

menggunakan `leadingIcon`, label input, dan opsi keyboard khusus angka. Fungsi ini dipanggil di `CalculatorTipApp()`.

Pada baris ke 135-208 itu mendefinisikan fungsi `roundTheTipRow()` untuk menampilkan teks dan komponen `Switch` (saklar) yang dapat mengatur apakah jumlah tip dibulatkan atau tidak. Komponen ini diletakkan dalam `Row` dengan padding dan alignment horizontal yang sesuai.Lalu mendefinisikan fungsi `calculateTip()` untuk menghitung jumlah tip berdasarkan input pengguna. Perhitungan dilakukan dengan mengalikan nominal tagihan dengan persen tip. Jika `roundUp` bernilai true, maka hasil tip dibulatkan ke atas menggunakan `ceil`. Hasil akhirnya diformat menjadi format mata uang lokal Amerika Serikat.

Kemudian mendefinisikan fungsi `DefaultPreview()` sebagai fungsi preview bawaan dari Jetpack Compose untuk menampilkan tampilan UI dalam Android Studio saat sedang dikembangkan, tanpa perlu menjalankan aplikasi di emulator atau perangkat.
Kemudian Mendefinisikan fungsi `TipPercentageDropdown()` sebagai komponen dropdown menu untuk memilih persen tip (15%, 18%, 20%). Digunakan `ExposedDropdownMenuBox` dan `ExposedDropdownMenu` dari Material 3. State `expanded` digunakan untuk menampilkan atau menyembunyikan dropdown, dan komponen `TextField` digunakan untuk menampilkan nilai yang dipilih. Ketika pengguna memilih salah satu opsi, dropdown akan menutup secara otomatis dan nilai terpilih akan dikirim ke parameter `onOptionSelected`.

SOAL 2

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

XML merupakan metode atau cara tradisuional atau imperatif yang telah digunakan untuk membuat UI, metode ini bersifat imperatif, untuk elemen elemen UI dituliskan

dalam file terpisah dengan file nama XML, kemudian dihubungkan dengan file activity atau fragment. Kelebihan menggunakan XML antara lain adalah kestabilannya karena metode ini sudah matang dan luas digunakan, dokumentasi yang melimpah, dan kemudahan dalam memisahkan logika dan tampilan, tapi ada beberapa kekurangan untuk XML seperti struktur code yang sangat susah dan ribet, sering sekali struktur kodenya panjang-panjang, dalam menangani UI yang dinamis kurang fleksibel dan performanya seoptimal metode baru yaitu Jetpack Compose.

Jetpack Compose merupakan metode modern dari google berbasis bahasa kotlin, Compose bersifat deklaratif karena tidak dipisah seperti XML, kelebihannya lebih fleksibel dan ringkas, serta secara alami mendukung UI bersifat reaktifm, sehinga sangat cocok digunakan pada aplikasi yang memerlukan perubahan tampilan jika ada perubahan data. Coompose juga dengan baik dengan fitur fitur modern. Kelebihan lainnya adalah kemudahan dalam membentuk komponen UI costum tanpa memerlukan banyak kode tambahan meski sangat fleksibel cepat dan ringkas, Jetpack Compose baru dan beberapa fiturnya belum stabil.

MODUL 3: BUILD A SCROLLABLE LIST

SOAL 1

Soal Praktikum:

1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:

   List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose) 2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas 3.

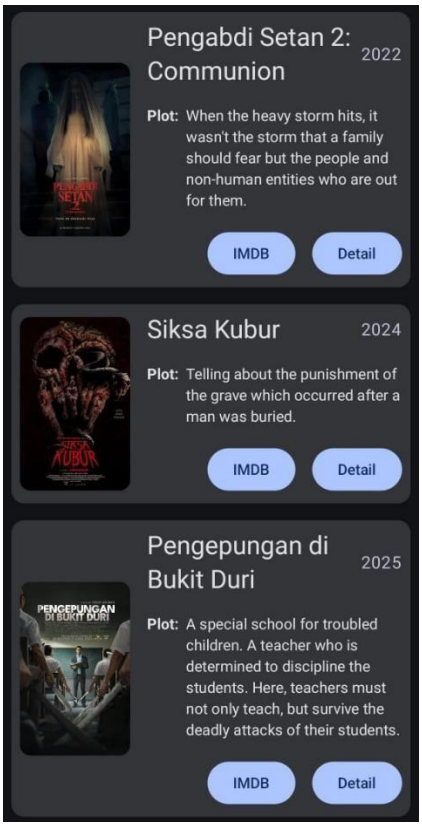
   Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah 4. Terdapat 2 button dalam list, dengan fungsi berikut:

a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item
c. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
d. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
e. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)

   2. Aplikasi berbasis XML harus menggunakan ViewBinding

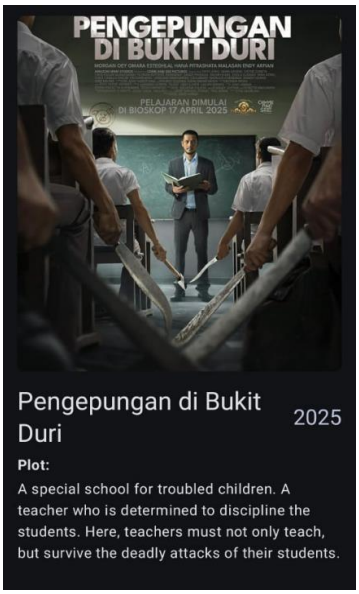
   Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

   UI item list harus berisi 1 gambar, 2 button (intent eksplisit dan navigasi), dan 2 baris teks dan setiap baris memiliki 2 teks yang berbeda. Diusahakan agar desain UI item list menyerupai UI berikut:



Gambar 16 Soal 1

Desain UI laman detail bebas, tetapi diusahakan untuk mengikuti kaidah desain Material Design dan data item ditampilkan penuh di laman detail seperti contoh berikut:



Gambar 17 Soal 1

A. **Source Code XML**

MainActivity.kt

Table 9 Source Code Jawaban Soal 1

```
1  package com.example.mobilelegendcharacterlistxml
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5
6
7  class MainActivity : AppCompatActivity() {
8
9
```

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

        val fragmentManager = supportFragmentManager
        val homeFragment = HomeFragment()
        val                fragment                =
fragmentManager.findFragmentByTag(HomeFragment::cla
ss.java.simpleName)
        if (fragment !is HomeFragment) {
            fragmentManager
                .beginTransaction()
                .add(R.id.frame_container,
homeFragment, HomeFragment::class.java.simpleName)
                .commit()
        }
    }
}
```

HeroML.kt :

Table 10 Source Code Jawaban Soal 1

```
package com.example.mobilelegendcharacterlistxml
import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
data class HeroML(
    val name: String,
```

```
    val image: Int,
    val url: String,
    val description: String
): Parcelable
```

HeroMLAdapter.kt:

Table 11 Source Code Jawaban Soal 1

```
package com.example.mobilelegendcharacterlistxml

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import
com.example.mobilelegendcharacterlistxml.databindin
g.ItemCharMlBinding

class HeroMLAdapter(
    private val listHero: ArrayList<HeroML>,
    private val onDetailClick: (String) -> Unit,
    private val onPenjelasanClick: (String, Int,
String) -> Unit
)                                                   :
RecyclerView.Adapter<HeroMLAdapter.ListViewHolder>(
) {

    inner class ListViewHolder(val binding:
ItemCharMlBinding) :
        RecyclerView.ViewHolder(binding.root) {
        fun bind(character: HeroML) {
```

```
            binding.tvItemName.text = character.name

binding.imgItemMl.setImageResource(character.image)
            binding.tvIsi.text                        =
character.description


binding.btnDescription.setOnClickListener {
                onDetailClick(character.url)
            }


binding.btnPenjelasan.setOnClickListener {
                onPenjelasanClick(character.name,
character.image, character.description)
            }
        }
    }

    override     fun     onCreateViewHolder(parent:
ViewGroup, viewType: Int): ListViewHolder {
        val                binding                =
ItemCharMlBinding.inflate(LayoutInflater.from(paren
t.context), parent, false)
        return ListViewHolder(binding)
    }

    override fun getItemCount(): Int = listHero.size


```

```
53      override fun onBindViewHolder(holder:
54  ListViewHolder, position: Int) {
55          holder.bind(listHero[position])
56      }
57  }
```

HomeFragment.kt:

Table 12 Source Code Jawaban Soal 1

```
 1  package com.example.mobilelegendcharacterlistxml
 2
 3  import android.content.Intent
 4  import android.net.Uri
 5  import android.os.Bundle
 6  import androidx.fragment.app.Fragment
 7  import android.view.LayoutInflater
 8  import android.view.View
 9  import android.view.ViewGroup
10  import
11  androidx.recyclerview.widget.LinearLayoutManager
12  import
13  com.example.mobilelegendcharacterlistxml.databindi
14  ng.FragmentHomeBinding
15
16  class HomeFragment : Fragment() {
17
18      private var _binding: FragmentHomeBinding? =
19  null
20      private val binding get() = _binding!!
21
```

```
    private lateinit var characterAdapter:
HeroMLAdapter
    private val list = ArrayList<HeroML>()


    override fun onCreateView(
        inflater: LayoutInflater, container:
ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding =
FragmentHomeBinding.inflate(inflater, container,
false)

        list.clear()
        list.addAll(getListHeroML())
        setupRecyclerView()

        return binding.root

    }

    private fun setupRecyclerView() {
        characterAdapter = HeroMLAdapter(
            list,
            onDetailClick = { url ->
                val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(url))
                startActivity(intent)
            },
```

```
            onPenjelasanClick = { name, image,
description ->
                val detailFragment =
DetailFragment().apply {
                    arguments = Bundle().apply {
                        putString("EXTRA_NAME",
name)
                        putInt("EXTRA_PHOTO",
image)

putString("EXTRA_DESCRIPTION", description)
                    }
                }


parentFragmentManager.beginTransaction()
                    .replace(R.id.frame_container,
detailFragment)
                    .addToBackStack(null)
                    .commit()
            }
        )

        binding.rvCharacter.apply {
            layoutManager =
LinearLayoutManager(context)
            adapter = characterAdapter
            setHasFixedSize(true)
        }
    }
```

```
 82
 83     private fun getListHeroML(): ArrayList<HeroML>
 84 {
 85         val                dataName                =
 86 resources.getStringArray(R.array.data_name)
 87         val               dataPhoto                =
 88 resources.obtainTypedArray(R.array.data_photo)
 89         val                dataLink                =
 90 resources.getStringArray(R.array.data_link)
 91         val                dataDesc                =
 92 resources.getStringArray(R.array.data_desc)
 93         val listCharacterML = ArrayList<HeroML>()
 94         for (i in dataName.indices) {
 95             val              character             =
 96 HeroML(dataName[i],dataPhoto.getResourceId(i,     -
 97 1),dataLink[i],dataDesc[i])
 98             listCharacterML.add(character)
 99         }
100         dataPhoto.recycle()
101         return listCharacterML
102     }
103
104     override fun onDestroyView() {
105         super.onDestroyView()
106         _binding = null
107     }
108 }
109
```

DetailFragment.kt:

Table 13 Source Code Jawaban Soal 1

```
1  package com.example.mobilelegendcharacterlistxml
2
3  import android.os.Bundle
4  import androidx.fragment.app.Fragment
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
8  import
9  com.example.mobilelegendcharacterlistxml.databindi
10 ng.FragmentDetailBinding
11
12
13 class DetailFragment : Fragment() {
14
15     private var _binding: FragmentDetailBinding? =
16 null
17     private val binding get() = _binding!!
18
19
20     override fun onCreateView(
21         inflater:    LayoutInflater,    container:
22 ViewGroup?,
23         savedInstanceState: Bundle?
24     ): View {
25         _binding                                    =
26 FragmentDetailBinding.inflate(inflater,  container,
27 false)
28
29
```

```
30          val                    name                    =
31  arguments?.getString("EXTRA_NAME")
32          val                   photo                    =
33  arguments?.getInt("EXTRA_PHOTO")
34          val               description                  =
35  arguments?.getString("EXTRA_DESCRIPTION")
36
37
38          binding.tvItemName.text = name
39          photo?.let {
40              binding.imgItemMl.setImageResource(it)
41              binding.tvIsi.text = description
42          }
43
44          return binding.root
45      }
46
47      override fun onDestroyView() {
48          super.onDestroyView()
49          _binding = null
50      }
    }
```

activity_main.xml:

Table 14 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout

xmlns:android="http://schemas.android.com/apk/res/
android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/frame_container"
    tools:context=".MainActivity">
</FrameLayout>
```

fragment_detail.xml:

Table 15 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/a
ndroid"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-
auto"
    tools:context=".DetailFragment">

    <ImageView
        android:id="@+id/img_item_ml"
```

```
10          android:layout_width="100dp"
11          android:layout_height="150dp"
            android:layout_marginTop="84dp"
12          android:scaleType="centerCrop"
13          app:layout_constraintEnd_toEndOf="parent"
14  app:layout_constraintStart_toStartOf="parent"
15          app:layout_constraintTop_toTopOf="parent"
16          tools:src="@tools:sample/avatars" />
17      <TextView
18          android:id="@+id/tv_item_name"
            android:layout_width="wrap_content"
19          android:layout_height="wrap_content"
20          android:layout_marginTop="20dp"
21          app:layout_constraintEnd_toEndOf="parent"
22  app:layout_constraintStart_toStartOf="parent"
23          android:textSize="30dp"
24  app:layout_constraintTop_toBottomOf="@+id/img_item_
25  ml"
26          tools:text="Nama Chara" />
27      <TextView
28          android:id="@+id/tv_isi"
            android:layout_width="0dp"
29          android:layout_height="wrap_content"
30          android:layout_marginTop="32dp"
            android:text="Deskripsi"
```

```
31          android:textSize="16sp"
32          app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="1.0"
33
34  app:layout_constraintStart_toStartOf="parent"
35
    app:layout_constraintTop_toBottomOf="@+id/tv_item_n
36  ame" />
37
38  </androidx.constraintlayout.widget.ConstraintLayout
39  >
40
```

fragment_home.xml:

Table 16 Source Code Jawaban Soal 1

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <androidx.constraintlayout.widget.ConstraintLayout
 3
    xmlns:android="http://schemas.android.com/apk/res/
 4  android"
        xmlns:tools="http://schemas.android.com/tools"
 5      xmlns:app="http://schemas.android.com/apk/res-
 6  auto"
 7      android:layout_width="match_parent"
        android:layout_height="match_parent"
 8      tools:context=".HomeFragment">
 9
```

```
10      <!-- TODO: Update blank fragment layout -->
11      <androidx.recyclerview.widget.RecyclerView
            android:layout_width="0dp"
12          android:layout_height="0dp"
13          android:id="@+id/rv_character"
14          android:layout_margin="15dp"

15  app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintEnd_toEndOf="parent"

17  app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
    />
19

20  </androidx.constraintlayout.widget.ConstraintLayou
21  t>
```

item_char_ml.xml:

Table 17 Source Code Jawaban Soal 1

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/
 3  android"
 4
    xmlns:card_view="http://schemas.android.com/apk/re
 5  s-auto"
 6      xmlns:tools="http://schemas.android.com/tools"
 7      android:id="@+id/card_view"
        android:layout_width="match_parent"
```

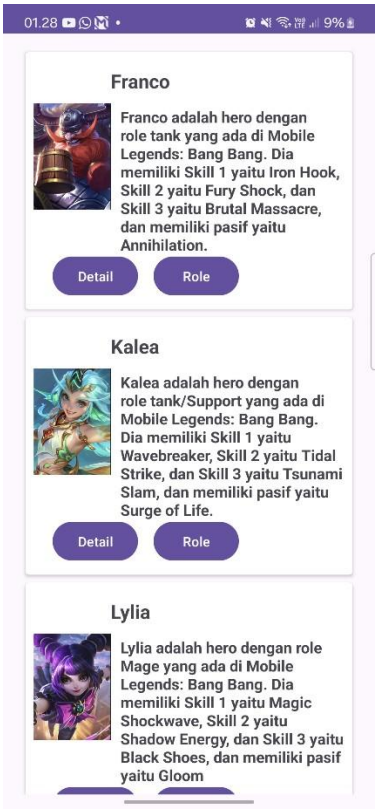
```
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginStart="8dp"
    android:layout_marginTop="4dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="4dp"
    card_view:cardCornerRadius="4dp">



<androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="266dp"
        android:padding="8dp">

        <ImageView
            android:id="@+id/img_item_ml"
            android:layout_width="80dp"
            android:layout_height="110dp"
            android:scaleType="centerCrop"

card_view:layout_constraintBottom_toTopOf="@+id/bt
n_description"

card_view:layout_constraintStart_toStartOf="parent
"

card_view:layout_constraintTop_toTopOf="parent" />


```

```
29          <TextView
30              android:id="@+id/tv_item_name"
31              android:layout_width="0dp"
32              android:layout_height="wrap_content"
33              android:layout_marginTop="8dp"
34              android:textSize="20sp"
35              android:textStyle="bold"
36
card_view:layout_constraintBottom_toTopOf="@+id/tv
_isi"
37
card_view:layout_constraintEnd_toEndOf="parent"
38
card_view:layout_constraintStart_toEndOf="@id/img_
39  item_ml"
40
card_view:layout_constraintTop_toTopOf="parent"
41
42  card_view:layout_constraintVertical_bias="0.0"
43              tools:text="Nama Hero" />
44
45          <Button
46              android:id="@+id/btn_description"
47              android:layout_width="wrap_content"
48              android:layout_height="wrap_content"
49              android:layout_marginTop="164dp"
                android:layout_marginStart="20dp"
                android:text="Detail"
```

```
50
51  card_view:layout_constraintStart_toStartOf="parent
52  "
53
54  card_view:layout_constraintTop_toBottomOf="@id/tv_
55  item_name" />
56
57          <Button
58              android:id="@+id/btn_penjelasan"
59              android:layout_width="wrap_content"
60              android:layout_height="wrap_content"
61              android:layout_marginStart="16dp"
62              android:layout_marginTop="164dp"
63              android:text="Role"
64
65  card_view:layout_constraintStart_toEndOf="@+id/btn
66  _description"
67
68  card_view:layout_constraintTop_toBottomOf="@id/tv_
69  item_name" />
70
71          <TextView
72              android:id="@+id/tv_isi"
73              android:layout_width="0dp"
74              android:layout_height="wrap_content"
75              android:layout_marginStart="10dp"
76              android:layout_marginTop="60dp"
77              android:textSize="16sp"
78              android:textStyle="bold"
79
```

```
80
81  card_view:layout_constraintEnd_toEndOf="parent"
82  card_view:layout_constraintHorizontal_bias="1.0"
83
84  card_view:layout_constraintStart_toEndOf="@+id/img
    _item_ml"
85
86  card_view:layout_constraintTop_toTopOf="parent"
87  tools:text="jsndjnjkjnsjnsjidnjksniaunsijnjniskjna
88  ksjnkjnkdjnkajsnkjdhisnkjniaanskdnsihaksnidjhaksni
    dsjaksndisnaksjndisjnaksnjaklnjianskaniusndsj" />
89
90
    </androidx.constraintlayout.widget.ConstraintLayou
    t>
        </androidx.cardview.widget.CardView>
```

B. **Output Program**



Gambar 18 Screenshot Hasil Jawaban Soal 1

Gambar 19 Screenshot Hasil Jawaban Soal 1

C. **Pembahasan**

MainActivity.kt:

Berikut adalah penjelasan kode yang kamu berikan dengan format yang sama seperti contohmu, sudah diberi jarak antarbagiannya, dan ditambah catatan jika ada hal yang perlu disiapkan di `build.gradle`:

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 3-4, dilakukan import terhadap `android.os.Bundle` yang digunakan untuk menyimpan dan meneruskan data antar lifecycle `Activity`, serta

`androidx.appcompat.app.AppCompatActivity`, yaitu superclass dari activity yang mendukung fitur-fitur kompatibilitas ke versi Android lama menggunakan AndroidX.

Pada line 6–17, didefinisikan class `MainActivity` yang merupakan turunan dari `AppCompatActivity`, yaitu activity utama pada aplikasi ini. Di dalam method `onCreate`, pertama-tama dipanggil `super.onCreate(savedInstanceState)` untuk memanggil implementasi superclass dan menginisialisasi activity. Kemudian, dipanggil `setContentView(R.layout.activity_main)` untuk menetapkan layout XML utama activity menggunakan file `activity_main.xml` sebagai tampilan UI.

Masih di dalam `onCreate`, objek `fragmentManager` diambil dari `supportFragmentManager`, yang digunakan untuk mengelola fragment dalam aplikasi. Sebuah instance dari `HomeFragment` dibuat dan disimpan dalam variabel `homeFragment`. Kemudian dilakukan pengecekan apakah fragment dengan tag `HomeFragment::class.java.simpleName` belum ditambahkan. Jika belum, maka fragment `HomeFragment` ditambahkan ke dalam layout dengan ID `frame_container` menggunakan `FragmentTransaction` dan ditandai dengan tag yang sama.

HeroML.kt

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 2, diimpor `android.os.Parcelable`, yaitu interface yang digunakan untuk mengirim objek custom antar komponen Android (seperti antar Activity atau Fragment) melalui `Intent` atau `Bundle`.

Pada line 3, diimpor `kotlinx.parcelize.Parcelize`, yaitu annotation yang digunakan untuk menyederhanakan implementasi interface `Parcelable` tanpa perlu menulis kode boilerplate seperti `writeToParcel()` dan `describeContents()` secara manual.

Pada line 5–11, dideklarasikan data class `HeroML` yang menampung data karakter Mobile

Legends. Kelas ini menggunakan anotasi `@Parcelize` untuk menggunakan `@Parcelize` harus mengubah gradle plugins `{id 'kotlin-parcelize'}` agar bisa otomatis diubah menjadi parcelable object. Properti-properti di dalam class ini terdiri dari:

- `name` bertipe `String` untuk menyimpan nama hero
- `image` bertipe `Int` untuk menyimpan ID resource gambar di drawable
- `url` bertipe `String` untuk menyimpan link yang berkaitan dengan hero tersebut
- `description` bertipe `String` untuk menyimpan deskripsi hero

Class ini mengimplementasikan `Parcelable` agar objeknya dapat dikirim lewat `Intent` atau disimpan dalam `Bundle` saat berpindah antar Fragment atau Activity.

HeroMLAdapter.kt

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 3–6, dilakukan import terhadap beberapa komponen penting yaitu

`LayoutInflater` untuk mengubah file XML layout menjadi objek View, `ViewGroup` sebagai parent dari layout item di RecyclerView, serta `RecyclerView` dari library AndroidX yang berfungsi menampilkan daftar data secara efisien dan fleksibel. Kemudian di line 6, diimpor `ItemCharMlBinding`, yaitu kelas yang secara otomatis dihasilkan oleh ViewBinding berdasarkan file layout `item_char_ml.xml`. Kelas binding ini memudahkan akses ke view dalam layout XML tanpa perlu memanggil `findViewById()` secara manual, viewbinding ini perlu untuk menambah di gradle yaitu `viewBinding {enabled = true}`.

Pada line 8–11, didefinisikan class `HeroMLAdapter` yang merupakan turunan dari

`RecyclerView.Adapter`. Adapter ini digunakan untuk mengatur tampilan daftar hero Mobile Legends. Konstruktor adapter menerima tiga parameter: `listHero`, yaitu daftar objek `HeroML` yang ingin ditampilkan; `onDetailClick`, yaitu lambda function yang dijalankan saat tombol "Description" diklik dan membawa data berupa `String` (URL hero); serta `onPenjelasanClick`, yaitu lambda function yang dijalankan saat tombol

"Penjelasan" diklik dan membawa data berupa `String`, `Int`, dan `String` (nama, ID gambar, dan deskripsi hero).

Pada line 13–23, dideklarasikan inner class `ListViewHolder` yang mewarisi `RecyclerView.ViewHolder`. Kelas ini berisi fungsi `bind()` yang bertugas mengikat data dari objek `HeroML` ke view dalam layout. Komponen UI seperti `tvItemName`, `imgItemMl`, dan `tvIsi` masing-masing diatur untuk menampilkan nama hero, gambar hero menggunakan ID dari resource, serta deskripsi hero. Di dalam fungsi yang sama, listener ditambahkan pada dua tombol yaitu `btnDescription` dan `btnPenjelasan`.

Tombol `btnDescription` akan memicu lambda `onDetailClick` dengan parameter URL hero, sedangkan tombol `btnPenjelasan` akan memicu lambda `onPenjelasanClick` dengan parameter nama, ID gambar, dan deskripsi hero tersebut.

Pada line 25–28, fungsi `onCreateViewHolder()` bertugas membuat instance dari `ListViewHolder`. Layout `item_char_ml.xml` di-*inflate* menggunakan `ItemCharMlBinding` untuk membuat tampilan setiap item pada RecyclerView. Binding ini kemudian digunakan untuk membentuk objek `ListViewHolder` yang akan merepresentasikan satu item hero.

Pada line 30, fungsi `getItemCount()` mengembalikan jumlah total data dalam `listHero`. Nilai ini menentukan berapa banyak item yang akan ditampilkan oleh RecyclerView.

Pada line 32–34, fungsi `onBindViewHolder()` dipanggil oleh RecyclerView untuk menampilkan data pada posisi tertentu. Fungsi ini akan mengambil data `HeroML` dari `listHero` berdasarkan indeks posisi, lalu memanggil fungsi `bind()` pada ViewHolder untuk menampilkannya ke dalam UI item tersebut.

HomeFragment.kt

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 3–11, dilakukan import terhadap berbagai komponen penting untuk membangun fragment, seperti `Intent`, `Uri`, `Bundle`, `Fragment`, `View`, `LayoutInflater`, `ViewGroup`, dan `LinearLayoutManager` dari

`RecyclerView`. Selain itu, juga diimpor `FragmentHomeBinding` yang merupakan kelas ViewBinding otomatis dari layout XML `fragment_home.xml`. Namun, baris import `android.R.attr.description`, `android.R.attr.name`, dan

`android.system.Os.link` sebetulnya **tidak diperlukan dan dapat dihapus**, karena tidak digunakan dalam kode dan bisa menimbulkan kebingungan karena berasal dari resource bawaan Android, bukan dari proyek aplikasi ini.

Pada line 13–16, didefinisikan class `HomeFragment` yang merupakan turunan dari `Fragment`. Fragment ini berfungsi menampilkan daftar karakter Mobile Legends dalam bentuk list menggunakan `RecyclerView`. Properti `_binding` digunakan sebagai tempat menyimpan binding terhadap layout, yang kemudian diakses aman melalui properti `binding`. Adapter untuk RecyclerView dideklarasikan dalam `characterAdapter`, dan daftar data hero dikelola melalui `list`, yang merupakan `ArrayList<HeroML>`.

Pada line 18–27, override dilakukan terhadap fungsi `onCreateView()` untuk meng*inflate* layout fragment, mengisi daftar data hero dengan memanggil `getListHeroML()`, lalu menginisialisasi RecyclerView melalui fungsi `setupRecyclerView()`. Fungsi ini akan dijalankan saat tampilan fragment pertama kali dibuat, dan nilai kembaliannya adalah `binding.root`, yaitu root dari layout hasil ViewBinding.

Pada line 29–53, didefinisikan fungsi `setupRecyclerView()` yang digunakan untuk mengatur komponen RecyclerView di dalam fragment. Di dalamnya, `characterAdapter` diinisialisasi menggunakan `HeroMLAdapter`, dan dua lambda function diberikan untuk menangani tombol klik pada setiap item: tombol "Description" akan membuka link URL hero melalui `Intent` dengan `ACTION_VIEW`, sedangkan tombol "Penjelasan" akan mengganti fragment saat ini dengan `DetailFragment`, sambil meneruskan data `name`, `image`, dan `description` melalui `Bundle`. Fragment baru ditampilkan menggunakan `parentFragmentManager` dengan metode `replace()` ke dalam `R.id.frame_container`, dan transaksi disimpan ke back stack agar bisa dikembalikan.

Pada line 55–64, fungsi `getListHeroML()` didefinisikan untuk mengambil data dari resource berupa array. Data diambil dari `strings.xml` (untuk nama, link, dan deskripsi hero) dan `arrays.xml/typedArray` (untuk gambar). Seluruh data

kemudian dikonversi menjadi list objek `HeroML`, lalu dikembalikan dalam bentuk `ArrayList`.

Pemanggilan `dataPhoto.recycle()` di akhir berfungsi untuk membebaskan resource yang sudah tidak digunakan.

Pada line 66–68, fungsi `onDestroyView()` di-override untuk menghindari memory leak pada Fragment. Binding dihapus dengan mengatur `_binding = null` saat view fragment dihancurkan, sesuai praktik yang direkomendasikan saat menggunakan ViewBinding di dalam Fragment.

DetailFragment.kt

Pada line 1, dideklarasikan nama package file Kotlin yaitu com.example.mobilelegendcharacterlistxml.

Pada line 3–9, dilakukan import terhadap beberapa komponen penting seperti Bundle, Fragment, View, ViewGroup, LayoutInflater, dan FragmentDetailBinding. Namun, import android.R.attr.text pada baris 3 sebetulnya **tidak diperlukan dan bisa dihapus**, karena tidak digunakan dalam kode dan justru berpotensi menimbulkan konflik dengan resource aplikasi sendiri. FragmentDetailBinding adalah kelas otomatis yang dibuat Android Studio berdasarkan layout XML fragment_detail.xml dan hanya bisa digunakan jika ViewBinding telah diaktifkan dalam file Gradle.

Pada line 11–14, didefinisikan class DetailFragment yang merupakan turunan dari Fragment. Fragment ini bertugas untuk menampilkan detail dari karakter Mobile Legends yang dipilih. Seperti konvensi umum dengan ViewBinding di Fragment, digunakan properti _binding sebagai nullable, dan binding sebagai non-nullable untuk akses aman terhadap elemen UI setelah onCreateView() dipanggil.

Pada line 16–27, fungsi onCreateView() di-override untuk meng-*inflate* layout fragment_detail.xml menggunakan FragmentDetailBinding. Kemudian, data yang dikirim melalui arguments (berupa nama, gambar, dan deskripsi hero) diambil menggunakan getString() dan getInt(). Data ini kemudian digunakan untuk mengisi tampilan: nama karakter dimasukkan ke TextView tvItemName, gambar diatur ke ImageView imgItemMl jika tidak null, dan deskripsi dimasukkan ke TextView tvIsi. Hal ini memastikan tampilan detail sesuai dengan hero yang dipilih sebelumnya di HomeFragment.

Pada line 29–32, fungsi onDestroyView() di-override untuk menghindari memory leak. Binding dihapus dengan mengatur _binding = null saat fragment dihancurkan. Ini adalah praktik yang direkomendasikan oleh Google saat menggunakan

ViewBinding di Fragment karena view Fragment memiliki siklus hidup yang berbeda dengan Fragment itu sendiri.

Activity_main.xml

Pada line 1–7, layout menggunakan `FrameLayout` dengan atribut `layout_width` dan `layout_height` diset ke `match_parent`, sehingga memenuhi seluruh layar perangkat. Elemen ini memiliki `id` yaitu `@+id/frame_container`, yang berfungsi sebagai referensi dalam `MainActivity.kt` untuk menambahkan fragment menggunakan metode `fragmentManager.beginTransaction().add(...)`.

Atribut `tools:context=".MainActivity"` hanya digunakan saat design preview di Android Studio untuk memberi tahu bahwa layout ini digunakan oleh kelas

`MainActivity`. Di dalam `FrameLayout` ini tidak ada elemen UI lain secara langsung karena kontennya akan diganti secara dinamis oleh fragment yang dimasukkan ke dalam `frame_container` tersebut saat runtime.

Item_char_ml.xml

File `item_char_ml.xml` merupakan layout yang digunakan sebagai tampilan item tunggal dalam `RecyclerView` di aplikasi ini. Layout ini digunakan di dalam `HeroMLAdapter.kt`, tepatnya di `ViewHolder` bernama `ListViewHolder`, untuk menampilkan setiap karakter Mobile Legends satu per satu dalam daftar.

Pada line 1–15, layout dibungkus oleh komponen `CardView` dari `androidx.cardview.widget.CardView`, yang memberikan efek bayangan dan sudut membulat pada item. Atribut `cardCornerRadius` diset ke 4dp agar sudutnya agak melengkung, dan terdapat margin di setiap sisi item agar tidak terlalu mepet satu sama lain ketika ditampilkan dalam daftar. `CardView` ini akan menjadi elemen visual utama dari setiap item hero.

Pada line 17–94, di dalam `CardView` terdapat `ConstraintLayout` yang digunakan untuk menyusun elemen-elemen UI secara fleksibel dengan constraint antar komponen. `ConstraintLayout` memiliki tinggi tetap 266dp dan padding 8dp agar isi tidak terlalu rapat ke tepi.

Di dalamnya, pertama ada `ImageView` dengan id `img_item_ml` (baris 19–26) yang digunakan untuk menampilkan gambar hero. Gambar ini berukuran 80dp x 110dp dan disetel dengan `scaleType="centerCrop"` agar gambar mengisi seluruh area dengan proporsional. Letaknya dikaitkan (`constraint`) di bagian atas layout dan sejajar secara vertikal dengan elemen lainnya.

Lalu ada `TextView` dengan id `tv_item_name` (baris 28–40) yang menampilkan nama hero. Lebarnya dibuat `0dp` karena menggunakan constraint start dan end, dengan ukuran teks 20sp dan gaya bold. Letaknya berada di atas elemen `tv_isi`, dan disesuaikan agar bersebelahan dengan `ImageView` sebelumnya.

Baris 42–54 Berikutnya terdapat dua buah `Button`, yaitu `btn_description` dan `btn_penjelasan`. Kedua tombol ini berada di bawah teks nama dan digunakan sebagai aksi untuk menampilkan link deskripsi hero (tombol Detail) dan penjelasan detail dalam fragment baru (tombol Role). Keduanya disejajarkan secara horizontal dengan sedikit jarak di antara keduanya.

Baris 56–68 Terakhir, ada `TextView` dengan id `tv_isi` yang berfungsi menampilkan deskripsi singkat dari hero tersebut. Komponen ini juga berada di samping `ImageView`, sejajar secara horizontal dan berada di atas tombol-tombol. Deskripsinya diset bold dengan ukuran 16sp agar mudah terbaca. Properti `tools:text` digunakan sebagai contoh isi deskripsi saat preview di Android Studio.

fragment_detail.xml

Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8. Pada line 2–5, digunakan `ConstraintLayout` sebagai root layout. Layout ini memungkinkan setiap elemen UI diposisikan relatif terhadap elemen lain dan/atau parentnya. Layout memiliki lebar dan tinggi `match_parent`, sehingga memenuhi seluruh layar. Tiga namespace juga dideklarasikan dalam elemen root: `xmlns:android` digunakan untuk atribut standar Android seperti `layout_width`, `id`, dan sebagainya; `xmlns:tools` digunakan untuk kebutuhan preview di Android Studio, seperti memberikan data dummy melalui `tools:text`; dan `xmlns:app` digunakan untuk atribut-atribut khusus dari ConstraintLayout, seperti `app:layout_constraintTop_toBottomOf`.

Pada line 7–14, didefinisikan sebuah `ImageView` dengan ID `img_item_ml`. Komponen ini digunakan untuk menampilkan gambar karakter Mobile Legends. Ukurannya diset sebesar 100dp x 150dp dan `scaleType` diatur `centerCrop`,

yang membuat gambar memenuhi seluruh area tampilan tanpa mengubah aspek rasio. Gambar ini diposisikan di tengah horizontal dengan margin atas sebesar 84dp dari parent layout.

Pada line 16–23, terdapat sebuah `TextView` dengan ID `tv_item_name` untuk menampilkan nama karakter. Ukuran teks dibuat cukup besar yaitu 30dp agar nama karakter lebih menonjol di layar. Elemen ini diletakkan tepat di bawah `ImageView` dengan margin atas 20dp, dan disejajarkan secara horizontal di tengah parent layout.

Pada line 25–32, dideklarasikan `TextView` kedua dengan ID `tv_isi`, yang digunakan untuk menampilkan deskripsi karakter Mobile Legends secara lebih lengkap. Lebarnya menggunakan `0dp` untuk mengikuti aturan `ConstraintLayout` (akan dihitung berdasarkan batas kiri dan kanan), dengan tinggi menyesuaikan isi. Posisi elemen ini berada di bawah nama karakter dengan margin atas 32dp. Ukuran teks disesuaikan agar tetap nyaman dibaca, yaitu 16sp, dan penempatan horizontalnya tetap berada di tengah lebar layout. fragment_char.xml Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8. Ini adalah deklarasi standar untuk file XML agar sistem dapat memahami format karakter yang digunakan.

Pada line 2–7, digunakan `ConstraintLayout` sebagai root layout. `ConstraintLayout` merupakan jenis layout fleksibel yang memungkinkan setiap elemen UI diposisikan secara relatif terhadap elemen lain maupun terhadap parent-nya. Layout ini memiliki lebar dan tinggi `match_parent`, yang berarti akan memenuhi seluruh ukuran layar. Tiga namespace juga didefinisikan di sini: Tiga namespace didefinisikan dalam elemen root `ConstraintLayout`. `xmlns:android` digunakan untuk atribut umum Android seperti `layout_width`, `id`, dan lainnya. `xmlns:tools` digunakan oleh Android Studio untuk keperluan preview layout, seperti menampilkan data dummy saat proses desain. Sementara itu, `xmlns:app` dipakai untuk atribut khusus yang berasal dari library AndroidX, termasuk atribut-atribut dari `ConstraintLayout` seperti `app:layout_constraintTop_toTopOf`.

Pada line 9–17, dideklarasikan sebuah komponen `RecyclerView` dengan ID `rv_character`. Komponen ini digunakan untuk menampilkan daftar karakter Mobile Legends dalam bentuk list atau grid yang bisa discroll. Lebar dan tinggi diatur `0dp`, yang berarti mengikuti aturan constraint dari `ConstraintLayout`. RecyclerView ini diberi margin sebesar 15dp di keempat sisi agar tidak menempel langsung ke tepi layar. Untuk posisi atributnya `app:layout_constraintTop_toTopOf="parent"` dan

`app:layout_constraintBottom_toBottomOf="parent"` digunakan untuk menempatkan komponen dari bagian atas hingga bawah, sehingga memenuhi tinggi parent-nya secara vertikal. Sedangkan `app:layout_constraintStart_toStartOf="parent"` dan `app:layout_constraintEnd_toEndOf="parent"` membuat `RecyclerView` memanjang secara horizontal dari kiri ke kanan, mengikuti lebar parent. Dengan keempat constraint ini, `RecyclerView` akan ditampilkan memenuhi seluruh layar, dengan margin di sekelilingnya yang telah ditentukan melalui atribut `android:layout_margin`.

A. Source Code Compose

MainActivity.kt:

Table 18 Source Code Jawaban Soal 1

```
 1  package com.example.mobilelegendcharacterlist
 2
 3  import android.os.Bundle
 4  import androidx.activity.ComponentActivity
 5  import androidx.activity.compose.setContent
 6  import androidx.activity.enableEdgeToEdge
 7  import androidx.compose.foundation.Image
 8  import androidx.compose.foundation.layout.*
 9  import androidx.compose.foundation.lazy.LazyColumn
10  import
11  androidx.compose.foundation.shape.RoundedCornerSh
12  ape
13  import androidx.compose.material3.*
14  import androidx.compose.runtime.Composable
15  import androidx.compose.ui.Alignment
16  import androidx.compose.ui.Modifier
17  import androidx.compose.ui.platform.LocalContext
```

```
18 import androidx.compose.ui.res.painterResource
19 import androidx.compose.ui.text.font.FontWeight
20 import androidx.compose.ui.text.style.TextOverflow
21 import androidx.compose.ui.tooling.preview.Preview
22 import androidx.compose.ui.unit.dp
23 import androidx.compose.ui.unit.sp
24 import
25 com.example.mobilelegendcharacterlist.ui.theme.Mo
26 bileLegendCharacterListTheme
27 import
28 com.example.mobilelegendcharacterlist.heroListML.
29 heroList
30 import android.content.Intent
31 import android.net.Uri
32 import androidx.navigation.NavHostController
33 import androidx.navigation.NavType
34 import androidx.navigation.compose.NavHost
35 import androidx.navigation.compose.composable
36 import
37 androidx.navigation.compose.rememberNavController
38 import androidx.navigation.navArgument
39
40 class MainActivity : ComponentActivity() {
41     override   fun   onCreate(savedInstanceState:
42 Bundle?) {
43         super.onCreate(savedInstanceState)
44         enableEdgeToEdge()
45         setContent {
46             MobileLegendCharacterListTheme {
47                 Surface(
```

```
48                      modifier                        =
49  Modifier.fillMaxSize(),
50                      color                           =
51  MaterialTheme.colorScheme.background
52                  ) {
53                      val       navController       =
54  rememberNavController()
55                      NavHost(
56                          navController               =
57  navController,
58                          startDestination            =
59  "heroList"
60                      ) {
61                          composable("heroList") {
62
63  HeroList(navController)
64                          }
65
66  composable("penjelasan/{description}/{image}",
67                              arguments = listOf(
68
69  navArgument("description")      {      type      =
70  NavType.StringType },
71
72  navArgument("image") { type = NavType.IntType }
73                              )
74                          ) { backStackEntry ->
75                              val    description    =
76  backStackEntry.arguments?.getString("description"
77  ) ?: ""
```

```
                                val       image       =
backStackEntry.arguments?.getInt("image") ?: 0

PenjelasanScreen(description, image)
                            }
                            }
                        }
                    }
                }
            }
        }

        @Composable
        fun HeroList(navController: NavHostController)
{
            LazyColumn(
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(20.dp)
            ) {
                items(heroList.size) { DataHero ->
                    val heroes = heroList[DataHero]
                    HeroItem(
                        name = heroes.name,
                        image = heroes.image,
                        url = heroes.url,
                        description                    =
heroes.description,
                        navController = navController
                    )
```

```
            }
        }
    }

    @Composable
    fun HeroItem(name: String, image: Int, url:
String, description: String, navController:
NavHostController) {
        val context = LocalContext.current
        Card(
            modifier = Modifier
                .fillMaxWidth()
                .padding(10.dp),
            shape = RoundedCornerShape(16.dp),
            elevation                          =
CardDefaults.cardElevation(defaultElevation     =
4.dp)
        ) {
            Row(
                modifier = Modifier
                    .padding(16.dp)
                    .fillMaxWidth(),
                verticalAlignment                =
Alignment.CenterVertically
            ) {
                Image(
                    painter = painterResource(id =
image),
                    contentDescription = null,
                    modifier = Modifier
```

```
                        .size(width    =    100.dp,
height = 120.dp)
                )
                Spacer(modifier                    =
Modifier.width(16.dp))
                Column(
                    modifier = Modifier
                        .weight(1f)
                ) {
                    Text(text = name, fontWeight =
FontWeight.Bold, fontSize = 20.sp)
                    Spacer(modifier                =
Modifier.height(4.dp))
                    Text(
                        text = description,
                        fontSize = 14.sp,
                        maxLines = 3,
                        overflow                   =
TextOverflow.Ellipsis
                    )
                    Spacer(modifier                =
Modifier.height(8.dp))
                    Row(
                        horizontalArrangement    =
Arrangement.SpaceBetween,
                        modifier = Modifier
                            .fillMaxWidth()

.wrapContentWidth(Alignment.Start),
                    ) {
```

```
168                         Button(
169                             onClick = {
170                                 val    intent    =
171 Intent(Intent.ACTION_VIEW, Uri.parse(url))
172 
173 context.startActivity(intent)
174                             },
175                             contentPadding       =
176 PaddingValues(horizontal = 16.dp, vertical = 8.dp),
177                             shape                =
178 RoundedCornerShape(50),
179                             modifier             =
180 Modifier.defaultMinSize(minWidth = 1.dp)
181                         ) {
182                             Text("Detail",
183 fontSize = 14.sp)
184                         }
185                         Spacer(modifier          =
186 Modifier.width(8.dp))
187                         Button(
188                             onClick = {
189                                 val  encodedDesc  =
190 Uri.encode(description)
191 
192 navController.navigate("penjelasan/${Uri.encode(d
193 escription)}/$image")
194                             },
195                             contentPadding       =
196 PaddingValues(horizontal = 16.dp, vertical = 8.dp),
197 
```

```
                                shape                 =
RoundedCornerShape(50),
                                modifier              =
Modifier.defaultMinSize(minWidth = 1.dp)
                            ) {
                                Text("Penjelasan",
fontSize = 13.sp)
                            }
                        }
                    }
                }
            }
        }

    @Composable
    fun   PenjelasanScreen(description:   String,
image: Int) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp)

.padding(WindowInsets.statusBars.asPaddingValues(
)),
            horizontalAlignment                 =
Alignment.CenterHorizontally
        ) {
            Image(
                painter  =  painterResource(id  =
image),
```

```
228                 contentDescription = null,
229                 modifier = Modifier
230                     .fillMaxWidth()
231                     .height(200.dp)
232             )
233             Spacer(modifier                         =
234 Modifier.height(16.dp))
235             Text(
236                 text = description,
237                 fontSize = 16.sp
238             )
239         }
240     }
241
242     @Preview(showBackground = true)
243     @Composable
244     fun GreetingPreview() {
245         MobileLegendCharacterListTheme {
246             Text("Preview List Hero")
247         }
248 }
```

dataList.kt:

Table 19 Source Code Jawaban Soal 1

```
1 package
2 com.example.mobilelegendcharacterlist.mobileLegend
3 DataList
4
5 data class DataHero(
6     val name: String,
      val image: Int,
      val url: String,
```

```
    val description : String
)
```

heroList.kt:

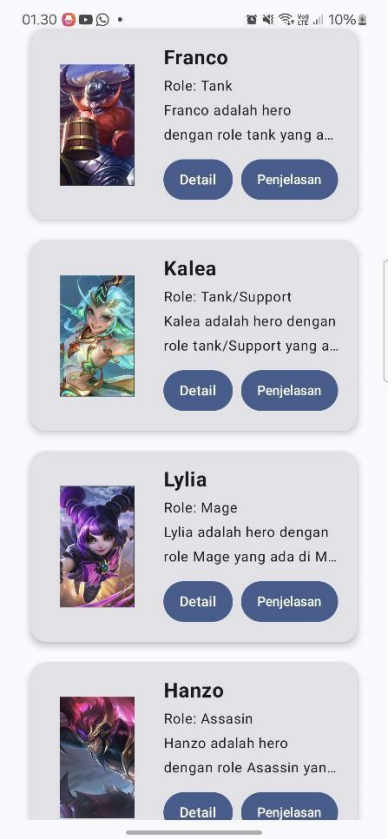
Table 20 Source Code Jawaban Soal 1

```
package
com.example.mobilelegendcharacterlist.heroListML



import com.example.mobilelegendcharacterlist.R
import
com.example.mobilelegendcharacterlist.mobileLegend
DataList.DataHero

val heroList = listOf(
    DataHero(name ="Franco",
        R.drawable.hero1,
        url =
"https://www.mobilelegends.com/hero/detail?channel
id=2678746&heroid=10",
        description = "Role: Tank\n" +
                "Franco adalah hero dengan role
tank yang ada di Mobile Legends: Bang Bang. Dia
memiliki Skill 1 yaitu 'Iron Hook', Skill 2 yaitu
'Fury Shock', dan Skill 3 yaitu 'Brutal Massacre',
dan memiliki pasif yaitu 'Annihilation'."),
    DataHero(name ="Kalea",
        R.drawable.hero2,


```

```
        url =
"https://www.mobilelegends.com/hero/detail?channel
id=2863075&heroid=128",
        description = "Role: Tank/Support\n" +
                "Kalea adalah hero dengan role
tank/Support yang ada di Mobile Legends: Bang
Bang. Dia memiliki Skill 1 yaitu 'Wavebreaker',
Skill 2 yaitu 'Tidal Strike', dan Skill 3 yaitu
'Tsunami Slam', dan memiliki pasif yaitu 'Surge of
Life'."),
    DataHero(name ="Lylia",
        R.drawable.hero3,
        url =
"https://www.mobilelegends.com/hero/detail?channel
id=2678822&heroid=86",
        description = "Role: Mage\n" +
                "Lylia adalah hero dengan role
Mage yang ada di Mobile Legends: Bang Bang. Dia
memiliki Skill 1 yaitu 'Magic Shockwave', Skill 2
yaitu 'Shadow Energy', dan Skill 3 yaitu 'Black
Shoes', dan memiliki pasif yaitu 'Angry Gloom'."),
    DataHero(name ="Hanzo",
        R.drawable.hero4,
        url =
"https://www.mobilelegends.com/hero/detail?channel
id=2678805&heroid=69",
        description = "Role: Assasin\n" +
                "Hanzo adalah hero dengan role
Asassin yang ada di Mobile Legends: Bang Bang. Dia
memiliki Skill 1 yaitu 'Ninjutsu: Demon Feast',
```

```
Skill 2 yaitu 'Ninjutsu: Dark Mist', dan Skill 3
yaitu 'Kinjutsu: Pinnacle Ninja', dan memiliki
pasif yaitu 'Ame no Habakiri'."),
    DataHero(name ="Lancelot",
        R.drawable.hero5,
        url =
"https://www.mobilelegends.com/hero/detail?channel
id=2678783&heroid=47",
        description = "Role: Assasin\n" +
                "Lancelot adalah hero dengan role
Assasin yang ada di Mobile Legends: Bang Bang. Dia
memiliki Skill 1 yaitu 'Puncture', Skill 2 yaitu
'Thorned Rose', dan Skill 3 yaitu 'Phantom
Execution', dan memiliki pasif yaitu 'Soul
Cutter'."),
    DataHero(name ="Lukas",
        R.drawable.hero6,
        url =
"https://www.mobilelegends.com/hero/detail?channel
id=2819992&heroid=127",
        description = "Role: Fighter\n" +
                "Lukas adalah hero dengan role
tank yang ada di Mobile Legends: Bang Bang. Dia
memiliki Skill 1 yaitu 'Flash Combo', Skill 2
yaitu 'Flash Step', dan Skill 3 yaitu 'Unleash the
Beast', dan memiliki pasif yaitu 'Hero's
Resolve'.")

)
```

B. Output Program



Gambar 20 Source Code Jawaban Soal 1

Gambar 21 Source Code Jawaban Soal 1

C. Pembahasan

MainActivity.kt:

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlist`.

Pada line 3–22, dilakukan import terhadap berbagai komponen dan library yang dibutuhkan dalam Jetpack Compose, seperti `Image`, `Column`, `Text`, `LazyColumn`, `Card`, `Button`, `TextOverflow`, dan `Modifier`. Selain itu juga mengimpor resource seperti `painterResource`, tools preview, serta dependensi untuk navigasi seperti `NavHostController`, `NavType`, dan `composable`. Ini memungkinkan kita untuk membangun UI deklaratif dan navigasi antar layar.

Pada line 24–42, didefinisikan class `MainActivity` yang merupakan turunan dari `ComponentActivity`, yaitu activity dasar untuk aplikasi yang menggunakan Jetpack Compose. Di dalam fungsi `onCreate()`, dipanggil `enableEdgeToEdge()` agar aplikasi tampil full screen pada device modern. Kemudian `setContent` digunakan untuk mengatur isi tampilan dari aplikasi, yang dibungkus dalam tema `MobileLegendCharacterListTheme`. Di dalamnya dibuat instance `NavHostController` dan didefinisikan `NavHost` dengan dua composable:

`heroList` untuk tampilan daftar hero, dan

`penjelasan/{description}/{image}` untuk menampilkan penjelasan hero berdasarkan parameter yang dikirim melalui navigasi.

Pada line 44–54, didefinisikan fungsi composable `HeroList()` yang menampilkan daftar hero menggunakan `LazyColumn`. Setiap item dalam list diambil dari `heroList` dan akan dirender menggunakan fungsi `HeroItem()`. Fungsi ini menerima `navController` sebagai parameter agar bisa melakukan navigasi ke halaman penjelasan saat tombol ditekan.

Pada line 56–98, didefinisikan fungsi composable `HeroItem()` yang menampilkan setiap karakter/hero dalam bentuk card. Di dalam card ditampilkan gambar, nama hero, dan deskripsi singkat yang dibatasi 3 baris (dengan `maxLines = 3` dan `overflow = TextOverflow.Ellipsis`). Terdapat dua tombol: tombol Detail yang membuka URL hero di browser menggunakan `Intent`, dan tombol Penjelasan yang menavigasi ke layar penjelasan dengan parameter `description` dan `image`. `Uri.encode()` digunakan untuk menghindari error saat ada karakter khusus di URL atau deskripsi.

Pada line 100–110, didefinisikan fungsi composable `PenjelasanScreen()` yang menerima dua parameter: `description` dan `image`. Di dalamnya ditampilkan gambar hero dan deskripsi lengkap yang ditampilkan dalam layout `Column` dengan padding serta responsif terhadap status bar (menggunakan `WindowInsets.statusBars.asPaddingValues()`).

Pada line 112–115, didefinisikan fungsi preview `GreetingPreview()` yang akan ditampilkan di Android Studio Preview. Ini hanya menampilkan `Text("Preview List Hero")` sebagai placeholder preview agar bisa melihat hasil tampilan saat sedang mengembangkan aplikasi tanpa harus menjalankan emulator.

dataList.kt

Pada line 1, dideklarasikan nama package yaitu `com.example.mobilelegendcharacterlist.mobileLegendDataList`,
yang merupakan sub-package dari aplikasi. Package ini berfungsi untuk mengelompokkan file-file yang berkaitan dengan data list hero Mobile Legends agar lebih terorganisasi.

Pada line 3, didefinisikan sebuah `data class` bernama `DataHero`.

`data class` di Kotlin secara otomatis menyediakan fungsi-fungsi penting seperti `toString()`, `equals()`, `hashCode()`, dan `copy()`, yang berguna untuk menyimpan dan mengelola data secara efisien.

Data class `DataHero` merepresentasikan satu entitas hero Mobile Legends dan memiliki empat properti:

Pada line 4, properti `name` bertipe `String`, menyimpan nama hero seperti "Franco", "Kalea", dll.

Pada line 5, properti `image` bertipe `Int`, menyimpan ID dari resource drawable (biasanya `R.drawable.nama_gambar`) yang digunakan untuk menampilkan gambar hero.

Pada line 6, properti `url` bertipe `String`, menyimpan tautan eksternal (link) resmi atau sumber informasi tentang hero tersebut, yang akan dibuka menggunakan browser ketika tombol "Detail" ditekan.

Pada line 7, properti `description` bertipe `String`, berisi penjelasan atau informasi lengkap mengenai hero, yang akan ditampilkan pada halaman `PenjelasanScreen`.

HeroMLAdapter.kt

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlist.heroListML`, yang menunjukkan bahwa file ini berada di dalam folder `heroListML` dari package utama aplikasi.

Pada line 3–5, dilakukan import terhadap resource dari file XML melalui `com.example.mobilelegendcharacterlist.R`, serta import class `DataHero` dari package `mobileLegendDataList`. Class `DataHero` kemungkinan besar didefinisikan sebagai data class yang merepresentasikan entitas hero Mobile Legends dengan properti seperti `name`, `imageResId`, `url`, dan `description`. Class ini berfungsi sebagai struktur data utama yang akan digunakan untuk menampilkan daftar hero pada tampilan RecyclerView atau Fragment detail.

Pada line 7–25, didefinisikan sebuah list bernama `heroList` yang berisi kumpulan objek `DataHero`. Setiap objek mewakili satu hero Mobile Legends dan memuat informasi lengkap seperti nama hero, ID gambar dari resource drawable (`R.drawable.hero1`, `hero2`, dan seterusnya), URL resmi untuk detail hero dari situs Mobile Legends, serta deskripsi singkat yang memuat role dan kemampuan dari hero tersebut. Deskripsi ditulis dengan format string multiline menggunakan karakter `\n` untuk membuat baris baru agar mudah dibaca di tampilan aplikasi. Seluruh data disimpan secara hardcoded dalam list menggunakan fungsi `listOf()`, sehingga data ini bersifat statis dan tidak dinamis dari API atau database.

Untuk dapat menggunakan gambar-gambar hero ini, kamu harus memastikan bahwa semua file gambar (`hero1.png`, `hero2.png`, dan seterusnya) sudah dimasukkan ke dalam folder `res/drawable` di project Android Studio kamu. Jika tidak, maka aplikasi akan mengalami error saat mencoba memuat resource tersebut.

SOAL 2

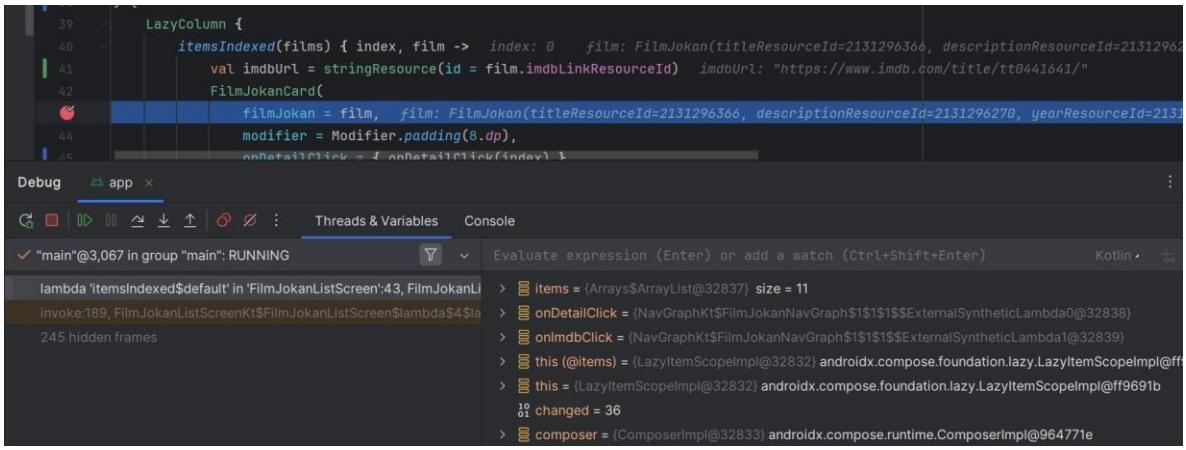
RecyclerView masih digunakan karena banyak aplikasi lama yang dibangun dengan View XML dan belum bermigrasi ke Jetpack Compose, RecyclerView menawarkan kontrol lebih mendetail terhadap tampilan list, seperti pengaturan layout, animasi, dan dekorasi item. Banyak library pihak ketiga juga masih bergantung pada RecyclerView. Meskipun LazyColumn di Jetpack Compose lebih ringkas dan modern, migrasi penuh memerlukan waktu dan sumber daya, sehingga RecyclerView tetap relevan di banyak proyek.

MODUL 4: VIEWMODEL AND DEBUGGING

SOAL 1

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

    a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
    b. Gunakan ViewModelFactory untuk membuat parameter dengan tipe data String di dalam ViewModel
    c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari
    ViewModel ke Fragment
    d. Install dan gunakan library Timber untuk logging event berikut:
        a. Log saat data item masuk ke dalam list
        b. Log saat tombol Detail dan tombol Explicit Intent ditekan
        c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
    e. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi.Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out



Gambar 22 Contoh Penggunaan Debugger

A. Source Code

MainActivity.kt

Table 21 Source Code Jawaban Soal 1 XML

```
1  package com.example.mobilelegendcharacterlistxml
2
3  import HomeFragment
4  import android.os.Bundle
5  import androidx.appcompat.app.AppCompatActivity
6
7  class MainActivity : AppCompatActivity() {
8
9      override fun onCreate(savedInstanceState:
10 Bundle?) {
11         super.onCreate(savedInstanceState)
12         setContentView(R.layout.activity_main)
13
```

```
        val fragmentManager = supportFragmentManager
        val homeFragment = HomeFragment()
        val                fragment                =
fragmentManager.findFragmentByTag(HomeFragment::cla
ss.java.simpleName)
        if (fragment !is HomeFragment) {
            fragmentManager
                .beginTransaction()
                .add(R.id.frame_container,
homeFragment, HomeFragment::class.java.simpleName)
                .commit()
        }
    }
}
```

HeroML.kt:

```
package
com.example.mobilelegendcharacterlistxml.model

import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
data class HeroML(
    val name: String,
    val image: Int,
    val url: String,
    val description: String
): Parcelable
```

HeroMLAdapter.kt :

```
1  package
2  com.example.mobilelegendcharacterlistxml.adapter
3
4  import android.view.LayoutInflater
5  import android.view.ViewGroup
6  import androidx.recyclerview.widget.RecyclerView
7  import
8  com.example.mobilelegendcharacterlistxml.databindin
9  g.ItemCharMlBinding
10 import
11 com.example.mobilelegendcharacterlistxml.model.Hero
12 ML
13
14 class HeroMLAdapter(
15     private val onDetailClick: (String) -> Unit,
16     private val onPenjelasanClick: (String, Int,
17 String) -> Unit,
18     private val logClick: (String) -> Unit
19 )                                                    :
20 RecyclerView.Adapter<HeroMLAdapter.ListViewHolder>(
21 ) {
22
23     private val data = ArrayList<HeroML>()
24
25     inner class ListViewHolder(val binding:
26 ItemCharMlBinding) :
27         RecyclerView.ViewHolder(binding.root) {
28         fun bind(character: HeroML) {
29             binding.tvItemName.text = character.name
```

```
22
23  binding.imgItemMl.setImageResource(character.image)
            binding.tvIsi.text                          =
24  character.description
25
26  binding.btnDescription.setOnClickListener {
27                  onDetailClick(character.url)
28              }
29
30
31  binding.btnPenjelasan.setOnClickListener {
32                  logClick(character.name)
33                  onPenjelasanClick(character.name,
34  character.image, character.description)
35              }
36          }
37      }
38
39      override      fun      onCreateViewHolder(parent:
40  ViewGroup, viewType: Int): ListViewHolder {
41          val                 binding                 =
42  ItemCharMlBinding.inflate(LayoutInflater.from(paren
43  t.context), parent, false)
44          return ListViewHolder(binding)
45      }
46
47      override fun getItemCount(): Int = data.size
48
49
```

```
50      override fun onBindViewHolder(holder:
51  ListViewHolder, position: Int) {
52          holder.bind(data[position])
53      }
54
55      fun submitList(list: List<HeroML>) {
56          data.clear()
57          data.addAll(list)
58          notifyDataSetChanged()
59      }
60  }
```

HomeFragment.kt

```
 1  import android.content.Intent
    import android.net.Uri
 2  import android.os.Bundle
    import android.view.LayoutInflater
 3  import android.view.View
    import android.view.ViewGroup
 4  import androidx.fragment.app.Fragment
    import androidx.lifecycle.ViewModelProvider
 5  import
    androidx.recyclerview.widget.LinearLayoutManager
 6  import
    com.example.mobilelegendcharacterlistxml.adapter.He
 7  roMLAdapter
    import
 8  com.example.mobilelegendcharacterlistxml.databindin
    g.FragmentHomeBinding
 9  import
    com.example.mobilelegendcharacterlistxml.ui.DetailF
    ragment
10  import
    com.example.mobilelegendcharacterlistxml.viewmodel.
11  HeroViewModel
    import
12  com.example.mobilelegendcharacterlistxml.viewmodel.
    HeroViewModelFactory
```

```
13 import com.example.mobilelegendcharacterlistxml.R
14 class HomeFragment : Fragment() {
15     private var _binding: FragmentHomeBinding? = null
       private val binding get() = _binding!!
16
       private lateinit var viewModel: HeroViewModel
17
       override fun onCreateView(
18         inflater:     LayoutInflater,     container:
ViewGroup?,
19         savedInstanceState: Bundle?
       ): View {
20         _binding                                    =
FragmentHomeBinding.inflate(inflater,     container,
false)
21         return binding.root
       }
22
       override    fun    onViewCreated(view:    View,
23 savedInstanceState: Bundle?) {
           super.onViewCreated(view,
24 savedInstanceState)
25         val                factory                 =
HeroViewModelFactory(requireActivity().application)
26         viewModel     =     ViewModelProvider(this,
factory)[HeroViewModel::class.java]
27
           val adapter = HeroMLAdapter(
28             onDetailClick = { url ->
                   val             intent             =
29 Intent(Intent.ACTION_VIEW, Uri.parse(url))
30                 startActivity(intent)
               },
31             onPenjelasanClick = { name, image, desc
32 ->
                   val bundle = Bundle().apply {
33                     putString("name", name)
34                     putInt("image", image)
                       putString("desc", desc)
35                 }
36
```

```
37                  val         detailFragment          =
DetailFragment().apply {
38                      arguments = bundle
39                  }
40
41  parentFragmentManager.beginTransaction()
42                      .replace(R.id.frame_container,
detailFragment)
43                      .addToBackStack(null)
44                      .commit()
45              },
46              logClick = { name ->
                    viewModel.logHeroClick(name)
47              }
        )
48
49          binding.rvCharacter.layoutManager           =
LinearLayoutManager(requireContext())
50          binding.rvCharacter.adapter = adapter
51
52  viewModel.heroList.observe(viewLifecycleOwner)      {
53  heroList ->
                adapter.submitList(heroList)
54          }
55      }
56
        override fun onDestroyView() {
57          super.onDestroyView()
58          _binding = null
        }
59  }
60
```

DetailFragment.kt:

```
1   package com.example.mobilelegendcharacterlistxml.ui
2
    import android.os.Bundle
    import android.view.LayoutInflater
```

```
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import
com.example.mobilelegendcharacterlistxml.databindin
g.FragmentDetailBinding

class DetailFragment : Fragment() {

    private var _binding: FragmentDetailBinding? =
null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater:     LayoutInflater,     container:
ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding                                    =
FragmentDetailBinding.inflate(inflater,   container,
false)

        val name = arguments?.getString("name")
        val photo = arguments?.getInt("image")
        val               description               =
arguments?.getString("desc")

        binding.tvItemName.text = name
        photo?.let {
            binding.imgItemMl.setImageResource(it)
        }
        binding.tvIsi.text = description

        return binding.root
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

HeroData.kt:

```
1  package com.example.mobilelegendcharacterlistxml.data
2
3  import android.content.Context
   import com.example.mobilelegendcharacterlistxml.R
4  import com.example.mobilelegendcharacterlistxml.model.HeroML
5
6  object HeroData {
7
8      fun getHeroList(context: Context): List<HeroML> {
9          val names = context.resources.getStringArray(R.array.data_name)
10         val descriptions = context.resources.getStringArray(R.array.data_desc)
11         val links = context.resources.getStringArray(R.array.data_link)
12
13         // Karena drawable tidak bisa diakses lewat string-array langsung, kita hardcode
14         val images = arrayOf(
15             R.drawable.hero1,
16             R.drawable.hero2,
17             R.drawable.hero3,
18             R.drawable.hero4,
19             R.drawable.hero5,
20             R.drawable.hero6
21         )
22
23         val list = ArrayList<HeroML>()
24         for (i in names.indices) {
25             val hero = HeroML(
26                 name = names[i],
27                 image = images[i],
                    url = links[i],
                    description = descriptions[i]
                )
                list.add(hero)
            }
            return list
        }
    }
```

HeroViewModel.kt:

```
1  package com.example.mobilelegendcharacterlistxml.viewmodel
2
3  import android.app.Application
4  import androidx.lifecycle.AndroidViewModel
5  import androidx.lifecycle.LiveData
6  import androidx.lifecycle.MutableLiveData
7  import com.example.mobilelegendcharacterlistxml.data.HeroData
8  import com.example.mobilelegendcharacterlistxml.model.HeroML
9  import timber.log.Timber
10
11 class HeroViewModel(application: Application) : AndroidViewModel(application) {
12
13     private val _heroList = MutableLiveData<List<HeroML>>()
14     val heroList: LiveData<List<HeroML>> = _heroList
15
16     init {
17         loadHeroData()
18         Timber.d("HeroViewModel initialized")
19     }
20
21     private fun loadHeroData() {
22         _heroList.value = HeroData.getHeroList(getApplication())
23         Timber.d("Data loaded: ${_heroList.value}")
24     }
25
26     fun logHeroClick(name: String) {
27         Timber.d("Klik hero: $name")
28     }
29 }
```

HeroViewModelFactory.kt:

```
1  package
   com.example.mobilelegendcharacterlistxml.viewmodel
2
3  import android.app.Application
   import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.ViewModelProvider
5
6  class HeroViewModelFactory(
       private val application: Application
   ): ViewModelProvider.NewInstanceFactory() {
7
8      override fun <T : ViewModel> create(modelClass:
   Class<T>): T {
9          if
   (modelClass.isAssignableFrom(HeroViewModel::class.j
10 ava)) {
               return HeroViewModel(application) as T
11         }
12         throw      IllegalArgumentException("Unknown
   ViewModel class")
13     }
   }
14
```

HeroApp.kt:

```
1  package com.example.mobilelegendcharacterlistxml
2
3  import android.app.Application
   import timber.log.Timber
4
5
6  class HeroApp : Application() {
       override fun onCreate() {
7          super.onCreate()
               Timber.plant(Timber.DebugTree())
8      }
9  }
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- Fragment container -->
    <FrameLayout
        android:id="@+id/frame_container"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>

```

item_char_ml.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/
android"

xmlns:card_view="http://schemas.android.com/apk/re
s-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/card_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginStart="8dp"
    android:layout_marginTop="4dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="4dp"
    card_view:cardCornerRadius="4dp">



<androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="266dp"
        android:padding="8dp">

        <ImageView
            android:id="@+id/img_item_ml"
            android:layout_width="80dp"
            android:layout_height="110dp"
            android:scaleType="centerCrop"
```

```
22
23  card_view:layout_constraintBottom_toTopOf="@+id/bt
24  n_description"
25
26  card_view:layout_constraintStart_toStartOf="parent
27  "
28
29  card_view:layout_constraintTop_toTopOf="parent" />
30
31
32          <TextView
33              android:id="@+id/tv_item_name"
34              android:layout_width="0dp"
35              android:layout_height="wrap_content"
36              android:layout_marginTop="8dp"
37              android:textSize="20sp"
38              android:textStyle="bold"
39
40  card_view:layout_constraintBottom_toTopOf="@+id/tv
41  _isi"
42
43  card_view:layout_constraintEnd_toEndOf="parent"
44
45  card_view:layout_constraintStart_toEndOf="@id/img_
46  item_ml"
47
48  card_view:layout_constraintTop_toTopOf="parent"
49
50  card_view:layout_constraintVertical_bias="0.0"
51              tools:text="Nama Hero" />
```

```
52
53          <Button
54              android:id="@+id/btn_description"
55              android:layout_width="wrap_content"
56              android:layout_height="wrap_content"
57              android:layout_marginTop="164dp"
58              android:layout_marginStart="20dp"
59              android:text="Detail"
60
61  card_view:layout_constraintStart_toStartOf="parent
62  "
63
64  card_view:layout_constraintTop_toBottomOf="@id/tv_
65  item_name" />
66
67          <Button
68              android:id="@+id/btn_penjelasan"
69              android:layout_width="wrap_content"
70              android:layout_height="wrap_content"
71              android:layout_marginStart="16dp"
72              android:layout_marginTop="164dp"
73              android:text="Role"
74
75  card_view:layout_constraintStart_toEndOf="@+id/btn
76  _description"
77
78  card_view:layout_constraintTop_toBottomOf="@id/tv_
79  item_name" />
80
81          <TextView
```

```
82              android:id="@+id/tv_isi"
83              android:layout_width="0dp"
84              android:layout_height="wrap_content"
85              android:layout_marginStart="10dp"
86              android:layout_marginTop="60dp"
87              android:textSize="16sp"
88              android:textStyle="bold"
89              android:maxLines="3"
90              android:ellipsize="end"
91
92  card_view:layout_constraintEnd_toEndOf="parent"
93
94  card_view:layout_constraintHorizontal_bias="1.0"
95
96  card_view:layout_constraintStart_toEndOf="@+id/img
97  _item_ml"
98
99  card_view:layout_constraintTop_toTopOf="parent"
100
    tools:text="jsndjnjkjnsjnsjidnjksniaunsijnjniskjna
101 ksjnkjnkdjnkajsnkjdhisnkjniaanskdnsihaksnidjhaksni
102 dsjaksndisnaksjndisjnaksnjaklnjianskaniusndsj" />
103
104
105 </androidx.constraintlayout.widget.ConstraintLayou
106 t>
107     </androidx.cardview.widget.CardView>
108
```

fragment_home.xml

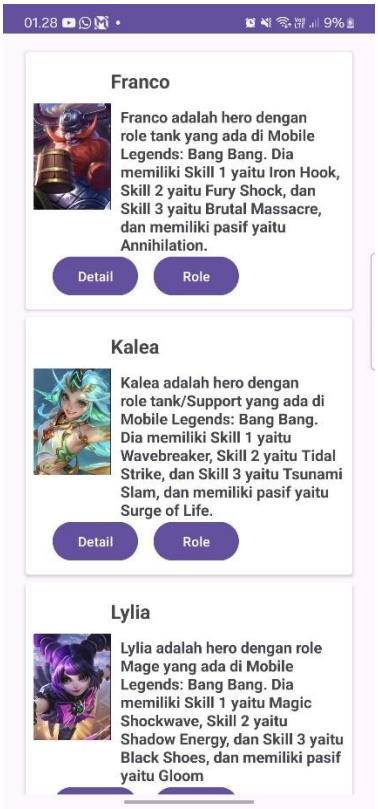
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.HomeFragment">

    <!-- TODO: Update blank fragment layout -->
    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:id="@+id/rv_character"
        android:layout_margin="15dp"

app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

fragment_detail.xml:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/
3  android"
4      xmlns:tools="http://schemas.android.com/tools"
       android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      xmlns:app="http://schemas.android.com/apk/res-
   auto"
7      tools:context=".ui.DetailFragment">
8
9      <ImageView
           android:id="@+id/img_item_ml"
10         android:layout_width="100dp"
11         android:layout_height="150dp"
12         android:layout_marginTop="84dp"
           android:scaleType="centerCrop"
13         app:layout_constraintEnd_toEndOf="parent"
14
   app:layout_constraintStart_toStartOf="parent"
15         app:layout_constraintTop_toTopOf="parent"
16         tools:src="@tools:sample/avatars" />
17
       <TextView
18         android:id="@+id/tv_item_name"
19         android:layout_width="wrap_content"
           android:layout_height="wrap_content"
20         android:layout_marginTop="20dp"
21         app:layout_constraintEnd_toEndOf="parent"
```

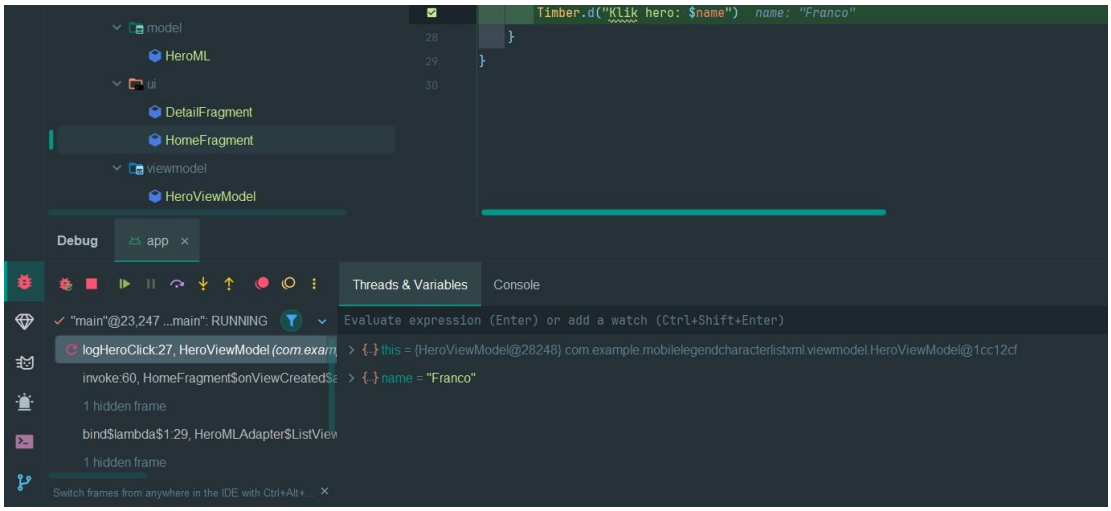
```
22
23  app:layout_constraintStart_toStartOf="parent"
        android:textSize="30dp"
24
25  app:layout_constraintTop_toBottomOf="@+id/img_item
    _ml"
26      tools:text="Nama Chara" />
27
28      <TextView
29          android:id="@+id/tv_isi"
30          android:layout_width="0dp"
31          android:layout_height="wrap_content"
32          android:layout_marginTop="32dp"
33          android:text="Deskripsi"
34          android:textSize="16sp"
35          app:layout_constraintEnd_toEndOf="parent"
36          app:layout_constraintHorizontal_bias="1.0"
37
38  app:layout_constraintStart_toStartOf="parent"
39
40  app:layout_constraintTop_toBottomOf="@+id/tv_item_
41  name" />
42
43
44  </androidx.constraintlayout.widget.ConstraintLayou
45  t>
```

B. Output Program

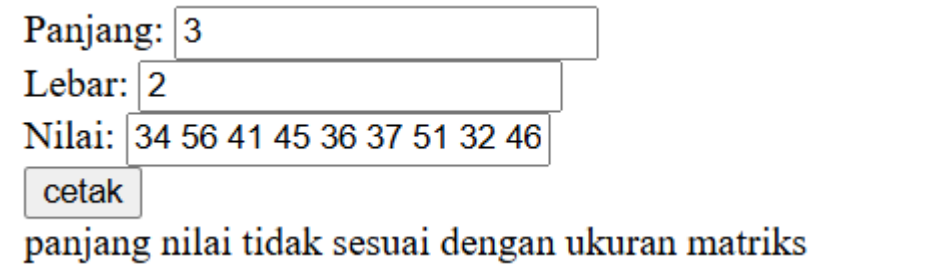


Gambar 23 Screenshot Hasil Jawaban Soal 1

Gambar 24 Screenshot Hasil Jawaban Soal 1 XML



Gambar 25 Screenshot Hasil Jawaban Soal 1 XML



Gambar 26 Screenshot Hasil Jawaban Soal 1

C. Pembahasan

MainActivity.kt:

Pada baris 1, `package com.example.mobilelegendcharacterlistxml` mendefinisikan nama package dari file Kotlin ini. Package ini merupakan struktur modular dalam proyek Android yang mengelompokkan file berdasarkan fungsinya.

Pada baris 3, `import HomeFragment` digunakan untuk mengimpor kelas

`HomeFragment` ke dalam file ini agar bisa digunakan sebagai tampilan awal pada saat aplikasi dijalankan.

Pada baris 4, `import android.os.Bundle` mengimpor class `Bundle` yang digunakan untuk menyimpan dan meneruskan data antar komponen, misalnya antar activity atau antar fragment.

Pada baris 5, `import androidx.appcompat.app.AppCompatActivity` mengimpor class `AppCompatActivity`, yaitu superclass dari activity yang menyediakan dukungan kompatibilitas untuk fitur-fitur Android modern pada perangkat lama.

Pada baris 8, `class MainActivity : AppCompatActivity()` mendefinisikan sebuah class bernama `MainActivity` yang merupakan turunan dari

`AppCompatActivity`, dan bertindak sebagai entry point utama saat aplikasi pertama kali dijalankan.

Pada baris 10, `override fun onCreate(savedInstanceState: Bundle?)`
adalah method yang dipanggil saat activity pertama kali dibuat. Method ini digunakan untuk menginisialisasi UI dan logika pertama kali.

Pada baris 11, `super.onCreate(savedInstanceState)` memanggil implementasi `onCreate` milik superclass (`AppCompatActivity`) untuk menjalankan logika dasar dari lifecycle activity.

Pada baris 12, `setContentView(R.layout.activity_main)` digunakan untuk menetapkan layout XML `activity_main.xml` sebagai tampilan utama dari activity ini.

Pada baris 14, `val fragmentManager = supportFragmentManager` mendeklarasikan sebuah variabel `fragmentManager` yang digunakan untuk mengelola fragment dalam activity, seperti menambahkan, mengganti, atau menghapus fragment.

Pada baris 15, `val homeFragment = HomeFragment()` membuat sebuah instance baru dari class `HomeFragment` yang nantinya akan digunakan untuk ditampilkan sebagai tampilan utama aplikasi.

Pada baris 16, `val fragment = fragmentManager.findFragmentByTag(HomeFragment::class.java.simpleName)` digunakan untuk mencari fragment yang telah ditambahkan sebelumnya berdasarkan tag-nya, untuk menghindari duplikasi fragment saat konfigurasi berubah (seperti rotasi layar).

Pada baris 17, `if (fragment !is HomeFragment)` adalah kondisi untuk memeriksa apakah fragment yang sedang aktif bukan instance dari `HomeFragment`.

Pada baris 18–20, jika kondisi terpenuhi, maka akan dilakukan transaksi fragment menggunakan `fragmentManager.beginTransaction()`, lalu `add()` untuk menambahkan `homeFragment` ke dalam `frame_container`, dan `commit()` untuk menjalankan transaksi tersebut.

HeroML.kt

Pada baris 1, `package com.example.mobilelegendcharacterlistxml.model` menyatakan bahwa file ini berada dalam package `model`, yang berfungsi untuk menyimpan class-class data (data model) dalam struktur proyek aplikasi. Pada baris 3, `import android.os.Parcelable` digunakan untuk mengimpor antarmuka `Parcelable`, yang memungkinkan objek dikirim antar komponen Android seperti `Intent` atau `Fragment`. Ini penting ketika kita ingin mengirim objek secara efisien melalui `Bundle`.

Pada baris 4, `import kotlinx.parcelize.Parcelize` digunakan untuk mengimpor anotasi `@Parcelize` dari Kotlin Android Extensions. Anotasi ini secara otomatis menghasilkan implementasi dari interface `Parcelable`, sehingga kita tidak perlu menuliskan kode `writeToParcel()` dan `describeContents()` secara manual.

Pada baris 6, anotasi `@Parcelize` digunakan di atas deklarasi data class untuk mengaktifkan fitur parcelisasi otomatis. Ini diperlukan agar objek `HeroML` dapat dengan mudah dikirim antar komponen Android, misalnya dari `HomeFragment` ke `DetailFragment`.

Pada baris 7–11, dideklarasikan sebuah `data class` bernama `HeroML` yang merepresentasikan data karakter Mobile Legends. Class ini memiliki empat properti:

- Pada baris 8, `val name: String` menyimpan nama dari karakter.
- Pada baris 9, `val image: Int` menyimpan ID resource gambar (misalnya `R.drawable.hero1`) dari karakter.
- Pada baris 10, `val url: String` menyimpan URL yang berisi link ke halaman detail karakter di internet.
- Pada baris 11, `val description: String` menyimpan deskripsi singkat mengenai karakter.

Class ini mengimplementasikan interface `Parcelable` melalui tanda titik dua `:` `Parcelable` pada akhir deklarasi class, sehingga objek `HeroML` dapat dikirim melalui `Intent`, `Bundle`, atau `arguments` dalam `Fragment`.

HeroMLAdapter.kt

Pada baris 1, `package com.example.mobilelegendcharacterlistxml.adapter` menyatakan bahwa file ini berada di dalam package `adapter`, yang biasanya digunakan untuk menyimpan class-class adapter yang berfungsi menghubungkan data dengan tampilan RecyclerView. Pada baris 3, `import android.view.LayoutInflater` digunakan untuk mengkonversi layout XML ke dalam objek View. Layout ini nantinya akan digunakan sebagai item pada RecyclerView. Pada baris 4, `import android.view.ViewGroup` mengimpor class `ViewGroup` yang menjadi induk dari item layout yang akan ditampilkan.

Pada baris 5, `import androidx.recyclerview.widget.RecyclerView` mengimpor class RecyclerView dan komponen terkaitnya untuk menampilkan daftar item yang dapat discroll. Pada baris 6, `import com.example.mobilelegendcharacterlistxml.databinding.ItemChar MlBinding` mengimpor class binding yang otomatis dibuat dari file layout `item_char_ml.xml`. Binding ini digunakan untuk mengakses elemen-elemen di layout item tanpa perlu menggunakan `findViewById`. Pada baris 7, `import com.example.mobilelegendcharacterlistxml.model.HeroML` mengimpor model data `HeroML` yang akan ditampilkan di RecyclerView.

Pada baris 9, dideklarasikan class `HeroMLAdapter` yang merupakan turunan dari `RecyclerView.Adapter`. Class ini menerima tiga parameter lambda:

- Pada baris 10, `onDetailClick: (String) -> Unit` akan dipanggil saat tombol detail ditekan, dan menerima URL sebagai argumen.
- Pada baris 11, `onPenjelasanClick: (String, Int, String) ->`

`Unit` akan dipanggil saat tombol penjelasan ditekan, dan menerima nama, gambar, dan deskripsi sebagai parameter.

- Pada baris 12, `logClick: (String) -> Unit` digunakan untuk mencatat log saat tombol penjelasan ditekan.

Pada baris 14, `data` dideklarasikan sebagai `ArrayList<HeroML>` untuk menyimpan daftar hero yang akan ditampilkan. Data ini akan diisi melalui method `submitList()`.

Pada baris 16–29, class `ListViewHolder` dideklarasikan sebagai inner class dari adapter. Class ini menerima parameter `binding` bertipe `ItemCharMlBinding` dan bertanggung jawab untuk mengikat data ke tampilan setiap item RecyclerView. Pada baris 17, konstruktor dipanggil dengan `binding.root` sebagai root View.

Pada baris 18–29, method `bind()` digunakan untuk mengatur data ke dalam View pada satu item:

- Pada baris 19, `binding.tvItemName.text = character.name` menampilkan nama karakter.
- Pada baris 20, `binding.imgItemMl.setImageResource(character.image)` menampilkan gambar karakter berdasarkan resource ID.
- Pada baris 21, `binding.tvIsi.text = character.description` menampilkan deskripsi singkat karakter.

Pada baris 23, listener untuk tombol `btnDescription` diatur. Saat tombol ditekan, lambda `onDetailClick` dipanggil dengan parameter URL dari karakter. Pada baris 26, listener untuk tombol `btnPenjelasan` diatur. Pertama, lambda `logClick` dipanggil untuk mencatat interaksi pengguna berdasarkan nama karakter. Lalu, lambda `onPenjelasanClick` dipanggil dengan parameter nama, gambar, dan deskripsi karakter.

Pada baris 31–34, method `onCreateViewHolder()` digunakan untuk membuat instance dari ViewHolder. Pada baris 32, layout `ItemCharMlBinding` di-inflate dari XML menggunakan `LayoutInflater`. Pada baris 33, objek ViewHolder baru dikembalikan.

Pada baris 36, method `getItemCount()` mengembalikan jumlah item dalam daftar data. Fungsi ini digunakan oleh RecyclerView untuk menentukan berapa banyak item yang akan ditampilkan.

Pada baris 38, method `onBindViewHolder()` bertanggung jawab untuk memanggil fungsi `bind()` pada ViewHolder dan memberikan data yang sesuai berdasarkan posisi.

Pada baris 41–44, method `submitList()` digunakan untuk meng-update data di dalam adapter. Pada baris 42, data lama dihapus menggunakan `data.clear()`. Pada baris 43, data baru ditambahkan menggunakan `data.addAll(list)`. Pada baris 44, `notifyDataSetChanged()` dipanggil untuk memberitahu RecyclerView agar memperbarui seluruh tampilan data.

HomeFragment.kt

Pada baris 1, `import android.content.Intent` digunakan untuk mengimpor class `Intent`, yang berfungsi untuk melakukan navigasi atau membuka aplikasi lain seperti browser. Pada baris 2, `import android.net.Uri` digunakan untuk mengimpor class `Uri`, yang dibutuhkan saat membuka link atau URL dari data karakter. Pada baris 3, `import android.os.Bundle` mengimpor class `Bundle`, yang digunakan untuk mengirim data antar Fragment. Pada baris 4, `import`

`android.view.LayoutInflater` digunakan untuk mengkonversi layout XML menjadi objek `View`. Pada baris 5, `import android.view.View` mengimpor class `View`, sebagai elemen dasar dari tampilan Android. Pada baris 6, `import android.view.ViewGroup` digunakan untuk menyatakan parent layout dari fragment yang di-inflate.

Pada baris 7, `import androidx.fragment.app.Fragment` mengimpor class `Fragment` dari AndroidX, yang digunakan sebagai dasar pembuatan `HomeFragment`. Pada baris 8, `import androidx.lifecycle.ViewModelProvider` digunakan untuk mendapatkan instance dari `ViewModel`, yaitu `HeroViewModel`, dengan lifecycleaware. Pada baris 9, `import`

`androidx.recyclerview.widget.LinearLayoutManager` digunakan untuk

menampilkan daftar data secara vertikal di dalam RecyclerView. Pada baris 10, `import`

`com.example.mobilelegendcharacterlistxml.adapter.HeroMLAdapter` mengimpor class adapter yang bertanggung jawab menghubungkan data `HeroML` dengan tampilan RecyclerView. Pada baris 11, `import`

`com.example.mobilelegendcharacterlistxml.databinding.FragmentHomeBinding` mengimpor class binding otomatis dari layout `fragment_home.xml` untuk memudahkan akses ke elemen UI tanpa `findViewById`.

Pada baris 12, `import` `com.example.mobilelegendcharacterlistxml.ui.DetailFragment` mengimpor class Fragment yang digunakan untuk menampilkan detail karakter Mobile Legends. Pada baris 13, `import`

`com.example.mobilelegendcharacterlistxml.viewmodel.HeroViewModel` mengimpor ViewModel yang menyimpan dan mengelola data list karakter. Pada baris 14, `import` `com.example.mobilelegendcharacterlistxml.viewmodel.HeroViewModelFactory` mengimpor factory yang digunakan untuk membuat instance `HeroViewModel` dengan parameter `Application`. Pada baris 15, `import` `com.example.mobilelegendcharacterlistxml.R` mengimpor class `R` yang mewakili semua resource dalam proyek seperti layout, drawable, dan ID view.

Pada baris 17, dideklarasikan class `HomeFragment` yang merupakan turunan dari `Fragment`. Pada baris 19, properti `_binding` bertipe nullable `FragmentHomeBinding` dideklarasikan untuk menghubungkan layout XML dengan kode Kotlin. Pada baris 20, `binding` didefinisikan sebagai non-nullable getter untuk menghindari kesalahan null pointer saat mengakses elemen UI.

Pada baris 22–26, method `onCreateView()` diimplementasikan untuk meng-inflate layout `fragment_home.xml` dan mengembalikan root view yang dihasilkan dari binding. Proses ini akan membuat tampilan Fragment berdasarkan layout XML.

Pada baris 28–56, method `onViewCreated()` digunakan untuk menginisialisasi data

dan tampilan setelah Fragment dan view-nya dibuat. Pada baris 30, `HeroViewModelFactory` diinisialisasi dengan `requireActivity().application` sebagai parameter, lalu digunakan untuk mendapatkan instance dari `HeroViewModel` dengan `ViewModelProvider`.

Pada baris 32–47, objek `HeroMLAdapter` diinisialisasi dengan tiga parameter lambda:

- Pada baris 33–35, lambda `onDetailClick` membuat `Intent` baru dengan `ACTION_VIEW` dan URI dari URL karakter, lalu membuka browser menggunakan `startActivity`.
- Pada baris 36–44, lambda `onPenjelasanClick` digunakan untuk memindahkan data karakter ke `DetailFragment`. Data disimpan dalam `Bundle`, lalu dimasukkan ke `arguments`. Fragment baru diatur dengan transaksi `replace`, kemudian ditambahkan ke back stack agar bisa kembali.
- Pada baris 45, lambda `logClick` memanggil fungsi `logHeroClick(name)` dari ViewModel untuk mencatat nama karakter yang ditekan melalui log.

Pada baris 50–51, `RecyclerView` diatur menggunakan `LinearLayoutManager` untuk menampilkan daftar karakter secara vertikal. Adapter yang telah dibuat juga dipasangkan ke RecyclerView melalui `binding.rvCharacter.adapter`.

Pada baris 53, `viewModel.heroList` diobservasi menggunakan `observe()` dengan lifecycle milik Fragment. Saat data dalam LiveData berubah, fungsi lambda akan dipanggil dan memanggil `adapter.submitList(heroList)` untuk memperbarui isi RecyclerView.

Pada baris 55–57, method `onDestroyView()` dipanggil saat view Fragment dihancurkan. Di sini, `_binding` diset `null` untuk menghindari memory leak karena referensi UI tidak lagi diperlukan.

DetailFragment.kt

Pada baris 1, `package com.example.mobilelegendcharacterlistxml.ui` menyatakan bahwa file ini berada di dalam package `ui`, yang biasanya digunakan untuk menyimpan class-class yang berhubungan dengan tampilan (user interface) dalam aplikasi.

Pada baris 3, `import android.os.Bundle` digunakan untuk mengimpor class `Bundle` yang berfungsi untuk menerima dan mengirim data antar Fragment atau Activity. Pada baris 4, `import android.view.LayoutInflater` digunakan untuk mengkonversi file XML layout menjadi objek `View`. Pada baris 5, `import android.view.View` mengimpor class `View` sebagai dasar dari semua komponen UI Android. Pada baris 6, `import android.view.ViewGroup` digunakan sebagai parent container dari layout yang akan di-inflate.

Pada baris 7, `import androidx.fragment.app.Fragment` mengimpor class `Fragment` dari AndroidX yang menjadi dasar dari class `DetailFragment`. Fragment ini digunakan untuk menampilkan halaman detail dari karakter Mobile Legends. Pada baris

8, `import com.example.mobilelegendcharacterlistxml.databinding.Fragment DetailBinding` mengimpor class binding yang secara otomatis dibuat dari layout `fragment_detail.xml`, yang digunakan untuk mengakses elemen UI tanpa harus menggunakan `findViewById`.

Pada baris 10, dideklarasikan class `DetailFragment` yang merupakan turunan dari `Fragment`. Class ini berfungsi untuk menampilkan tampilan detail dari salah satu karakter yang dipilih pada halaman utama.

Pada baris 12, variabel `_binding` bertipe nullable `FragmentDetailBinding` dideklarasikan untuk menyimpan instance binding dari layout. Pada baris 13, variabel `binding` dideklarasikan sebagai non-nullable dan digunakan untuk mengakses elemen layout secara aman selama fragment masih aktif.

Pada baris 15–24, method `onCreateView()` diimplementasikan untuk meng-inflate layout `fragment_detail.xml`, menampilkan data, dan mengembalikan root view-nya. Pada baris 16, binding diinisialisasi dengan `FragmentDetailBinding.inflate()` untuk mengakses elemen UI.

Pada baris 18–20, data dikirim dari Fragment sebelumnya diambil menggunakan `arguments?.getString()` dan `arguments?.getInt()`. Tiga data yang diambil adalah `name`, `image`, dan `desc`, yang mewakili nama karakter, resource ID gambar, dan deskripsi karakter.

Pada baris 22, `binding.tvItemName.text = name` digunakan untuk menampilkan nama karakter ke dalam TextView. Pada baris 23–25, jika `photo` tidak null, maka gambar karakter ditampilkan melalui `setImageResource()`. Pada baris 26, deskripsi karakter ditampilkan ke dalam TextView `tvIsi`.

Pada baris 28, `return binding.root` mengembalikan root view dari layout untuk ditampilkan sebagai isi dari Fragment.

Pada baris 30–32, method `onDestroyView()` dipanggil ketika view Fragment dihancurkan. Di dalam method ini, `_binding` diset `null` untuk menghindari memory leak, karena binding tidak lagi diperlukan setelah tampilan dihancurkan.

HeroData.kt:

Pada baris 1, `package com.example.mobilelegendcharacterlistxml.data` menyatakan bahwa file ini berada dalam package `data`, yang biasanya digunakan untuk menyimpan class atau

object yang berkaitan dengan pengolahan atau penyediaan data dalam aplikasi. Pada baris 3, `import android.content.Context` mengimpor class `Context` dari Android,

yang diperlukan untuk mengakses resource aplikasi seperti string-array. Pada baris 4,

`import com.example.mobilelegendcharacterlistxml.R` mengimpor class
`R` yang merepresentasikan semua resource dalam proyek, termasuk drawable dan stringarray. Pada baris 5, `import com.example.mobilelegendcharacterlistxml.model.HeroML` mengimpor data class `HeroML` yang akan digunakan untuk membuat daftar objek karakter.

Pada baris 7, dideklarasikan object `HeroData` yang merupakan singleton object di Kotlin. Object ini berfungsi sebagai penyedia data (data provider) untuk daftar karakter Mobile Legends yang akan ditampilkan pada aplikasi.

Pada baris 9, fungsi `getHeroList(context: Context): List<HeroML>` dideklarasikan dengan parameter `context` bertipe `Context`, dan akan mengembalikan list dari objek `HeroML`. Fungsi ini digunakan untuk mengambil data karakter dari resource dan menyusunnya menjadi objek `HeroML`.

Pada baris 10, `val names = context.resources.getStringArray(R.array.data_name)` digunakan
untuk mengambil array string `data_name` dari resource `strings.xml`, yang berisi daftar nama karakter. Pada baris 11, `val descriptions = context.resources.getStringArray(R.array.data_desc)` mengambil
array string yang berisi deskripsi dari setiap karakter. Pada baris 12, `val links = context.resources.getStringArray(R.array.data_link)` mengambil array string yang berisi URL link karakter.

Pada baris 14–21, karena ID drawable tidak dapat disimpan di dalam `strings.xml` atau `array.xml`, maka pada baris 15, array `images` didefinisikan secara manual (hardcode) menggunakan ID resource drawable. Setiap elemen dalam array ini mewakili gambar dari masing-masing karakter.

Pada baris 23, `val list = ArrayList<HeroML>()` mendeklarasikan list kosong bertipe `HeroML` yang nantinya akan diisi dengan data dari semua karakter. Pada baris 24, dilakukan perulangan `for (i in names.indices)` yang berarti perulangan sebanyak jumlah elemen di dalam array `names`.

Pada baris 25–28, setiap iterasi membuat objek `HeroML` menggunakan data dari array `names`, `images`, `links`, dan `descriptions` berdasarkan indeks yang sama. Pada baris 29, objek `hero` ditambahkan ke dalam list menggunakan `list.add(hero)`.

Pada baris 30, fungsi `getHeroList` mengembalikan seluruh list yang telah berisi objek `HeroML`.

HeroViewModel.kt:

Pada baris 1, `package com.example.mobilelegendcharacterlistxml.viewmodel` menyatakan
bahwa file ini berada di dalam package `viewmodel`, yang berfungsi untuk menyimpan kelas-kelas ViewModel dalam arsitektur MVVM (Model-View-ViewModel). Pada baris 3,

`import android.app.Application` mengimpor class `Application` dari Android, yang digunakan untuk mendapatkan konteks aplikasi dalam ViewModel. Pada baris 4, `import androidx.lifecycle.AndroidViewModel` mengimpor class `AndroidViewModel`, yaitu subclass dari `ViewModel` yang memiliki akses langsung ke `Application`. Ini diperlukan untuk mengambil resource seperti `Context`.

Pada baris 5, `import androidx.lifecycle.LiveData` digunakan untuk mengimpor class `LiveData`, yaitu komponen lifecycle-aware yang dapat diamati oleh UI. Pada baris 6, `import androidx.lifecycle.MutableLiveData` mengimpor versi mutable dari `LiveData`, yang digunakan untuk menyimpan dan memperbarui data. Pada baris 7, `import com.example.mobilelegendcharacterlistxml.data.HeroData` mengimpor class `HeroData`, yang berisi fungsi penyedia data karakter. Pada baris 8,`importcom.example.mobilelegendcharacterlistxml.model.HeroML` mengimpor data class `HeroML`, yang merepresentasikan data karakter Mobile Legends. Pada baris 9, `import timber.log.Timber` mengimpor library Timber yang digunakan untuk melakukan logging secara efisien.
Pada baris 11, dideklarasikan class `HeroViewModel` yang merupakan subclass dari `AndroidViewModel`. Class ini digunakan untuk mengelola dan menyimpan data karakter, serta bertahan saat rotasi layar. Konstruktor menerima parameter `application: Application`, yang diteruskan ke superclass `AndroidViewModel(application)`.

Pada baris 13, variabel `_heroList` bertipe `MutableLiveData<List<HeroML>>` digunakan untuk menyimpan list data karakter dan memungkinkan perubahan data. Pada baris 14, variabel `heroList` bertipe `LiveData<List<HeroML>>` hanya memberikan akses baca ke UI untuk mencegah perubahan langsung dari luar ViewModel.

Pada baris 16, blok `init` dipanggil secara otomatis saat `HeroViewModel` pertama kali dibuat. Pada baris 17, method `loadHeroData()` dipanggil untuk memuat data karakter
dari `HeroData`. Pada baris 18, `Timber.d("HeroViewModel initialized")` digunakan untuk mencatat bahwa ViewModel telah diinisialisasi, sebagai bagian dari proses debugging menggunakan Timber. ada baris 20–22, fungsi `loadHeroData()` digunakan untuk mengambil data karakter dari `HeroData` menggunakan `getHeroList(getApplication())`, karena `getHeroList` membutuhkan `Context`. Nilai yang didapat kemudian disimpan ke `_heroList.value`. Pada baris 22, `Timber.d("Data loaded: ${_heroList.value}")` digunakan untuk mencatat data yang berhasil dimuat ke log. Pada baris 24–26, fungsi `logHeroClick(name: String)` digunakan untuk mencatat nama karakter yang diklik ke dalam log menggunakan `Timber.d()`. Fungsi ini akan dipanggil dari Fragment saat tombol ditekan, sebagai bagian dari pencatatan aktivitas pengguna.

HeroViewModelFactory.kt:

Pada baris 1, package com.example.mobilelegendcharacterlistxml.viewmodel menyatakan bahwa file ini berada di dalam package viewmodel, yang digunakan untuk menyimpan class-class ViewModel dan komponennya dalam arsitektur MVVM. Pada baris 3, import android.app.Application mengimpor class Application dari Android yang digunakan untuk mendapatkan konteks global aplikasi. Pada baris 4, import androidx.lifecycle.ViewModel mengimpor class ViewModel sebagai superclass dasar untuk semua ViewModel yang digunakan dalam arsitektur lifecycle-aware. Pada baris 5, import androidx.lifecycle.ViewModelProvider mengimpor ViewModelProvider, yang digunakan untuk mengelola pembuatan dan penyimpanan instance ViewModel.

Pada baris 7, dideklarasikan class HeroViewModelFactory yang merupakan turunan dari ViewModelProvider.NewInstanceFactory. Class ini bertugas sebagai factory atau pembuat instance dari HeroViewModel, khususnya saat HeroViewModel membutuhkan parameter tambahan seperti Application.

Pada baris 8, variabel application bertipe Application dideklarasikan sebagai properti privat dari class, dan nilainya akan diberikan melalui konstruktor. Properti ini nantinya akan digunakan untuk menginisialisasi HeroViewModel.

Pada baris 10–14, method override fun <T : ViewModel> create(modelClass: Class<T>): T digunakan untuk membuat dan mengembalikan instance dari ViewModel. Pada baris 11, dilakukan pengecekan apakah modelClass merupakan

turunan dari HeroViewModel menggunakan isAssignableFrom(). Jika iya, maka pada baris 12, akan dikembalikan objek HeroViewModel(application) dan dikonversi menjadi tipe T. Jika modelClass bukan HeroViewModel, maka pada baris 14 akan dilemparkan IllegalArgumentException dengan pesan "Unknown ViewModel class" untuk menunjukkan bahwa kelas yang diminta tidak dikenali oleh factory ini.

Activity_main.xml

Pada line 1–7, layout menggunakan `FrameLayout` dengan atribut `layout_width` dan `layout_height` diset ke `match_parent`, sehingga memenuhi seluruh layar perangkat. Elemen ini memiliki `id` yaitu `@+id/frame_container`, yang berfungsi sebagai referensi dalam `MainActivity.kt` untuk menambahkan fragment menggunakan metode `fragmentManager.beginTransaction().add(...)`.

Atribut `tools:context=".MainActivity"` hanya digunakan saat design preview di Android Studio untuk memberi tahu bahwa layout ini digunakan oleh kelas `MainActivity`. Di dalam `FrameLayout` ini tidak ada elemen UI lain secara langsung karena kontennya akan diganti secara dinamis oleh fragment yang dimasukkan ke dalam `frame_container` tersebut saat runtime.

Item_char_ml.xml

File `item_char_ml.xml` merupakan layout yang digunakan sebagai tampilan item tunggal dalam `RecyclerView` di aplikasi ini. Layout ini digunakan di dalam `HeroMLAdapter.kt`, tepatnya di `ViewHolder` bernama `ListViewHolder`, untuk menampilkan setiap karakter Mobile Legends satu per satu dalam daftar.

Pada line 1–15, layout dibungkus oleh komponen `CardView` dari `androidx.cardview.widget.CardView`, yang memberikan efek bayangan dan sudut membulat pada item. Atribut `cardCornerRadius` diset ke 4dp agar sudutnya agak melengkung, dan terdapat margin di setiap sisi item agar tidak terlalu mepet satu sama lain ketika ditampilkan dalam daftar. `CardView` ini akan menjadi elemen visual utama dari setiap item hero.

Pada line 17–94, di dalam `CardView` terdapat `ConstraintLayout` yang digunakan untuk menyusun elemen-elemen UI secara fleksibel dengan constraint

antar komponen. `ConstraintLayout` memiliki tinggi tetap 266dp dan padding 8dp agar isi tidak terlalu rapat ke tepi.

Di dalamnya, pertama ada `ImageView` dengan id `img_item_ml` (baris 19–26) yang digunakan untuk menampilkan gambar hero. Gambar ini berukuran 80dp x 110dp dan disetel dengan `scaleType="centerCrop"` agar gambar mengisi seluruh area dengan proporsional. Letaknya dikaitkan (`constraint`) di bagian atas layout dan sejajar secara vertikal dengan elemen lainnya.

Lalu ada `TextView` dengan id `tv_item_name` (baris 28–40) yang menampilkan nama hero. Lebarnya dibuat `0dp` karena menggunakan constraint start dan end, dengan ukuran teks 20sp dan gaya bold. Letaknya berada di atas elemen `tv_isi`, dan disesuaikan agar bersebelahan dengan `ImageView` sebelumnya.

Baris 42–54 Berikutnya terdapat dua buah `Button`, yaitu `btn_description` dan `btn_penjelasan`. Kedua tombol ini berada di bawah teks nama dan digunakan sebagai aksi untuk menampilkan link deskripsi hero (tombol Detail) dan penjelasan detail dalam fragment baru (tombol Role). Keduanya disejajarkan secara horizontal dengan sedikit jarak di antara keduanya.

Baris 56–68 Terakhir, ada `TextView` dengan id `tv_isi` yang berfungsi menampilkan deskripsi singkat dari hero tersebut. Komponen ini juga berada di samping `ImageView`, sejajar secara horizontal dan berada di atas tombol-tombol. Deskripsinya diset bold dengan ukuran 16sp agar mudah terbaca. Properti `tools:text` digunakan sebagai contoh isi deskripsi saat preview di Android Studio.

fragment_detail.xml

Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8.

Pada line 2–5, digunakan `ConstraintLayout` sebagai root layout. Layout ini memungkinkan setiap elemen UI diposisikan relatif terhadap elemen lain dan/atau parentnya. Layout memiliki lebar dan tinggi `match_parent`, sehingga memenuhi seluruh layar. Tiga namespace juga dideklarasikan dalam elemen root: `xmlns:android` digunakan untuk atribut standar Android seperti `layout_width`, `id`, dan sebagainya; `xmlns:tools` digunakan untuk kebutuhan preview di Android Studio, seperti

memberikan data dummy melalui `tools:text`; dan `xmlns:app` digunakan untuk atribut-atribut khusus dari ConstraintLayout, seperti `app:layout_constraintTop_toBottomOf`.

Pada line 7–14, didefinisikan sebuah `ImageView` dengan ID `img_item_ml`. Komponen ini digunakan untuk menampilkan gambar karakter Mobile Legends. Ukurannya diset sebesar 100dp x 150dp dan `scaleType` diatur `centerCrop`, yang membuat gambar memenuhi seluruh area tampilan tanpa mengubah aspek rasio. Gambar ini diposisikan di tengah horizontal dengan margin atas sebesar 84dp dari parent layout.

Pada line 16–23, terdapat sebuah `TextView` dengan ID `tv_item_name` untuk menampilkan nama karakter. Ukuran teks dibuat cukup besar yaitu 30dp agar nama karakter lebih menonjol di layar. Elemen ini diletakkan tepat di bawah `ImageView` dengan margin atas 20dp, dan disejajarkan secara horizontal di tengah parent layout.

Pada line 25–32, dideklarasikan `TextView` kedua dengan ID `tv_isi`, yang digunakan untuk menampilkan deskripsi karakter Mobile Legends secara lebih lengkap. Lebarnya menggunakan `0dp` untuk mengikuti aturan `ConstraintLayout` (akan dihitung berdasarkan batas kiri dan kanan), dengan tinggi menyesuaikan isi. Posisi elemen ini berada di bawah nama karakter dengan margin atas 32dp. Ukuran teks disesuaikan agar tetap nyaman dibaca, yaitu 16sp, dan penempatan horizontalnya tetap berada di tengah lebar layout.

fragment_char.xml

Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8. Ini adalah deklarasi standar untuk file XML agar sistem dapat memahami format karakter yang digunakan.

Pada line 2–7, digunakan `ConstraintLayout` sebagai root layout.

`ConstraintLayout` merupakan jenis layout fleksibel yang memungkinkan setiap elemen UI diposisikan secara relatif terhadap elemen lain maupun terhadap parent-nya. Layout ini memiliki lebar dan tinggi `match_parent`, yang berarti akan memenuhi seluruh ukuran layar. Tiga namespace juga didefinisikan di sini:

Tiga namespace didefinisikan dalam elemen root `ConstraintLayout`. `xmlns:android` digunakan untuk atribut umum Android seperti `layout_width`, `id`, dan lainnya.

`xmlns:tools` digunakan oleh Android Studio untuk keperluan preview layout, seperti menampilkan data dummy saat proses desain. Sementara itu, `xmlns:app` dipakai untuk atribut khusus yang berasal dari library AndroidX, termasuk atribut-atribut dari `ConstraintLayout` seperti `app:layout_constraintTop_toTopOf`.

Pada line 9–17, dideklarasikan sebuah komponen `RecyclerView` dengan ID `rv_character`. Komponen ini digunakan untuk menampilkan daftar karakter Mobile Legends dalam bentuk list atau grid yang bisa discroll. Lebar dan tinggi diatur `0dp`, yang berarti mengikuti aturan constraint dari `ConstraintLayout`. RecyclerView ini diberi margin sebesar 15dp di keempat sisi agar tidak menempel langsung ke tepi layar. Untuk posisi atributnya `app:layout_constraintTop_toTopOf="parent"` dan `app:layout_constraintBottom_toBottomOf="parent"` digunakan untuk menempatkan komponen dari bagian atas hingga bawah, sehingga memenuhi tinggi parent-nya secara vertikal. Sedangkan `app:layout_constraintStart_toStartOf="parent"` dan `app:layout_constraintEnd_toEndOf="parent"` membuat `RecyclerView` memanjang secara horizontal dari kiri ke kanan, mengikuti lebar parent. Dengan keempat constraint ini, `RecyclerView` akan ditampilkan memenuhi seluruh layar, dengan margin di sekelilingnya yang telah ditentukan melalui atribut `android:layout_margin`.

A. Source Code Compose

MainActivity.kt

Table 22 Source Code Jawaban soal 1 Compose

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

```
 8 import
 9 androidx.compose.foundation.layout.fillMaxSize
10 import
11 androidx.lifecycle.viewmodel.compose.viewModel
12 import androidx.compose.material3.*
13 import androidx.compose.ui.Modifier
14 import
15 com.example.mobilelegendcharacterlist.ui.theme.Mobi
16 leLegendCharacterListTheme
17 import
18 com.example.mobilelegendcharacterlist.viewmodel.Her
19 oViewModel
20 import
21 com.example.mobilelegendcharacterlist.viewmodel.Her
22 oViewModelFactory
23 import
24 androidx.navigation.compose.rememberNavController
25 import androidx.navigation.compose.NavHost
26 import androidx.navigation.compose.composable
27 import androidx.navigation.NavType
28 import androidx.navigation.navArgument
29 import
30 com.example.mobilelegendcharacterlist.ui.screen.Det
31 ailScreen
32 import
33 com.example.mobilelegendcharacterlist.ui.screen.Hom
34 eScreen
35
36 class MainActivity : ComponentActivity() {
37
```

```
38      override fun onCreate(savedInstanceState:
39  Bundle?) {
40          super.onCreate(savedInstanceState)
41          enableEdgeToEdge()
42
43          setContent {
44              MobileLegendCharacterListTheme {
45                  val          navController          =
46  rememberNavController()
47
48                  val   viewModel:   HeroViewModel   =
49  viewModel(
50                      factory                        =
51  HeroViewModelFactory("List Hero ML")
52                  )
53
54                  Surface(
55                      modifier                       =
56  Modifier.fillMaxSize(),
57                      color                          =
58  MaterialTheme.colorScheme.background
59                  ) {
60                      NavHost(
61                          navController              =
62  navController,
63                          startDestination           =
64  "heroList"
65                      ) {
66                          composable("heroList") {
67
```

```
68                              HomeScreen(viewModel =
69 viewModel, navController = navController)
70                          }
71                          composable(
72                              route =
73 "penjelasan/{description}/{image}",
74                              arguments = listOf(
75
76 navArgument("description") { type =
77 NavType.StringType },
78
79 navArgument("image") { type = NavType.IntType }
80                              )
81                          ) { backStackEntry ->
82                              val description =
83 backStackEntry.arguments?.getString("description")
84 ?: ""
85                              val image =
86 backStackEntry.arguments?.getInt("image") ?: 0
87
88 DetailScreen(description = description, image =
89 image)
90                          }
91                      }
92                  }
93              }
          }
      }
}
```

DataHero.kt:

Table 23 Source Code Jawaban soal 1 Compose

```
1  package com.example.mobilelegendcharacterlist.model
2
3  data class DataHero(
4      val name: String,
5      val image: Int,
6      val url: String,
7      val description : String
8  )
```

HeroData.kt:

Table 24 Source Code Jawaban soal 1 Compose

```
 1  package com.example.mobilelegendcharacterlist.data
 2
 3  import com.example.mobilelegendcharacterlist.R
 4  import
 5  com.example.mobilelegendcharacterlist.model.DataH
 6  ero
 7
 8  object HeroData{
 9      val heroList = listOf(
10          DataHero(name ="Franco",
11              R.drawable.hero1,
12              url                                    =
13  "https://www.mobilelegends.com/hero/detail?channe
14  lid=2678746&heroid=10",
15              description = "Role: Tank\n" +
```

```
                    "Franco adalah hero dengan
role tank yang ada di Mobile Legends: Bang Bang.
Dia memiliki Skill 1 yaitu 'Iron Hook', Skill 2
yaitu 'Fury Shock', dan Skill 3 yaitu 'Brutal
Massacre',    dan    memiliki    pasif    yaitu
'Annihilation'."),
        DataHero(name ="Kalea",
            R.drawable.hero2,
            url                                 =
"https://www.mobilelegends.com/hero/detail?channe
lid=2863075&heroid=128",
            description = "Role: Tank/Support\n" +
                    "Kalea adalah hero dengan role
tank/Support yang ada di Mobile Legends: Bang Bang.
Dia memiliki Skill 1 yaitu 'Wavebreaker', Skill 2
yaitu 'Tidal Strike', dan Skill 3 yaitu 'Tsunami
Slam', dan memiliki pasif yaitu 'Surge of Life'."),
        DataHero(name ="Lylia",
            R.drawable.hero3,
            url                                 =
"https://www.mobilelegends.com/hero/detail?channe
lid=2678822&heroid=86",
            description = "Role: Mage\n" +
                    "Lylia adalah hero dengan role
Mage yang ada di Mobile Legends: Bang Bang. Dia
memiliki Skill 1 yaitu 'Magic Shockwave', Skill 2
yaitu 'Shadow Energy', dan Skill 3 yaitu 'Black
Shoes', dan memiliki pasif yaitu 'Angry Gloom'."),
        DataHero(name ="Hanzo",
            R.drawable.hero4,
```

```
            url                                         =
"https://www.mobilelegends.com/hero/detail?channe
lid=2678805&heroid=69",
            description = "Role: Assasin\n" +
                    "Hanzo adalah hero dengan role
Asassin yang ada di Mobile Legends: Bang Bang. Dia
memiliki Skill 1 yaitu 'Ninjutsu: Demon Feast',
Skill 2 yaitu 'Ninjutsu: Dark Mist', dan Skill 3
yaitu 'Kinjutsu: Pinnacle Ninja', dan memiliki
pasif yaitu 'Ame no Habakiri'."),
        DataHero(name ="Lancelot",
            R.drawable.hero5,
            url                                         =
"https://www.mobilelegends.com/hero/detail?channe
lid=2678783&heroid=47",
            description = "Role: Assasin\n" +
                    "Lancelot adalah hero dengan
role Assasin yang ada di Mobile Legends: Bang Bang.
Dia memiliki Skill 1 yaitu 'Puncture', Skill 2
yaitu 'Thorned Rose', dan Skill 3 yaitu 'Phantom
Execution', dan memiliki pasif yaitu 'Soul
Cutter'."),
        DataHero(name ="Lukas",
            R.drawable.hero6,
            url                                         =
"https://www.mobilelegends.com/hero/detail?channe
lid=2819992&heroid=127",
            description = "Role: Fighter\n" +
                    "Lukas adalah hero dengan role
tank yang ada di Mobile Legends: Bang Bang. Dia
```

```
76  memiliki Skill 1 yaitu 'Flash Combo', Skill 2 yaitu
77  'Flash Step', dan Skill 3 yaitu 'Unleash the
78  Beast', dan memiliki pasif yaitu 'Hero's
79  Resolve'.")

        )
    }
```

HomeScreen.kt:

Table 25 Source Code Jawaban soal 1 Compose

```
 1  package
 2  com.example.mobilelegendcharacterlist.ui.screen
 3
 4  import android.content.Intent
 5  import android.net.Uri
 6  import androidx.compose.foundation.Image
 7  import
 8  androidx.compose.foundation.layout.Arrangement
 9  import androidx.compose.foundation.layout.Column
10  import
11  androidx.compose.foundation.layout.PaddingValues
12  import androidx.compose.foundation.layout.Row
13  import androidx.compose.foundation.layout.Spacer
14  import
15  androidx.compose.foundation.layout.fillMaxWidth
16  import androidx.compose.foundation.layout.height
17  import androidx.compose.foundation.layout.padding
18  import androidx.compose.foundation.layout.size
```

```
19 import androidx.compose.foundation.layout.width
20 import
21 androidx.compose.foundation.layout.wrapContentWidt
22 h
23 import androidx.compose.foundation.lazy.LazyColumn
24 import
25 androidx.compose.foundation.shape.RoundedCornerSha
26 pe
27 import androidx.compose.material3.Button
28 import androidx.compose.material3.Card
29 import androidx.compose.material3.CardDefaults
30 import androidx.compose.material3.Text
31 import androidx.compose.runtime.Composable
32 import androidx.compose.runtime.collectAsState
33 import androidx.compose.ui.Alignment
34 import androidx.compose.ui.Modifier
35 import androidx.compose.ui.platform.LocalContext
36 import androidx.compose.ui.res.painterResource
37 import androidx.compose.ui.text.font.FontWeight
38 import androidx.compose.ui.text.style.TextOverflow
39 import androidx.compose.ui.unit.dp
40 import androidx.compose.ui.unit.sp
41 import androidx.navigation.NavController
42 import
43 com.example.mobilelegendcharacterlist.viewmodel.He
44 roViewModel
45 import timber.log.Timber
46
47 @Composable
48
```

```
49 fun      HomeScreen(viewModel:      HeroViewModel,
50 navController: NavController) {
51     val                  heroList                  =
52 viewModel.heroList.collectAsState().value
53     LazyColumn(
54         modifier = Modifier
55             .fillMaxWidth()
56             .padding(16.dp)
57     ) {
58         items(heroList.size) { index ->
59             val hero = heroList[index]
60             HeroCard(hero.name,         hero.image,
61 hero.url,     hero.description,     navController,
62 viewModel)
63
64         }
65     }
66 }
67
68 @Composable
69 fun HeroCard(
70     name: String,
71     image: Int,
72     url: String,
73     description: String,
74     navController: NavController,
75     viewModel: HeroViewModel
76 ) {
77     val context = LocalContext.current
78
```

```
 79      Card(
 80          modifier = Modifier
 81              .fillMaxWidth()
 82              .padding(8.dp),
 83          shape = RoundedCornerShape(12.dp),
 84          elevation                                    =
 85  CardDefaults.cardElevation(defaultElevation = 4.dp)
 86      ) {
 87          Row(
 88              modifier = Modifier
 89                  .padding(16.dp)
 90                  .fillMaxWidth(),
 91              verticalAlignment                        =
 92  Alignment.CenterVertically
 93          ) {
 94              Image(
 95                  painter   =   painterResource(id   =
 96  image),
 97                  contentDescription = name,
 98                  modifier = Modifier
 99                      .size(width = 100.dp, height =
100  120.dp)
101              )
102
103              Spacer(modifier                          =
104  Modifier.width(12.dp))
105
106              Column(
107                  modifier = Modifier.weight(1f)
108              ) {
```

```
109                 Text(text = name, fontWeight =
110 FontWeight.Bold, fontSize = 18.sp)
111                 Text(
112                     text = description,
113                     fontSize = 12.sp,
114                     maxLines = 3,
115                     overflow =
116 TextOverflow.Ellipsis
117                 )
118
119                 Spacer(modifier =
120 Modifier.height(8.dp))
121
122                 Row(
123                     modifier = Modifier
124                         .fillMaxWidth()
125                         .padding(top = 8.dp),
126                     horizontalArrangement =
127 Arrangement.Center
128                 ) {
129                     Button(
130                         onClick = {
131
132 viewModel.logHeroClick(name)
133                             val intent =
134 Intent(Intent.ACTION_VIEW, Uri.parse(url))
135
136 context.startActivity(intent)
137
138                         },
```

```
                        shape                       =
RoundedCornerShape(50),
                        contentPadding              =
PaddingValues(horizontal = 24.dp, vertical = 8.dp),
                        modifier = Modifier
                            .wrapContentWidth()
                    ) {
                        Text("Detail",  fontSize  =
14.sp)
                    }

                    Spacer(modifier                 =
Modifier.width(12.dp))

                    Button(
                        onClick = {

viewModel.logHeroClick(name)
                            val    encodedDesc    =
Uri.encode(description)

navController.navigate("penjelasan/${encodedDesc}/
$image")
                        },
                        shape                       =
RoundedCornerShape(50),
                        contentPadding              =
PaddingValues(horizontal = 16.dp, vertical = 8.dp),
                        modifier = Modifier
                            .wrapContentWidth()
```

```
169                     ) {
170                         Text("Penjelasan",
171 fontSize = 12.sp)
172                     }
173                 }
174             }
175         }
176     }
177 }
```

DetailScreen.kt:

Table 26 Source Code Jawaban soal 1 Compose

```
  1 package
  2 com.example.mobilelegendcharacterlist.ui.screen
  3
  4 import androidx.compose.foundation.Image
  5 import androidx.compose.foundation.layout.Column
  6 import androidx.compose.foundation.layout.Spacer
  7 import
  8 androidx.compose.foundation.layout.fillMaxSize
  9 import
 10 androidx.compose.foundation.layout.fillMaxWidth
 11 import androidx.compose.foundation.layout.height
 12 import androidx.compose.foundation.layout.padding
 13 import androidx.compose.material3.Text
 14 import androidx.compose.runtime.Composable
 15 import androidx.compose.ui.Alignment
 16 import androidx.compose.ui.Modifier
 17 import androidx.compose.ui.res.painterResource
```

```
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

@Composable
fun DetailScreen(description: String, image: Int) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        horizontalAlignment                          =
Alignment.CenterHorizontally
    ) {
        Image(
            painter = painterResource(id = image),
            contentDescription = null,
            modifier = Modifier
                .fillMaxWidth()
                .height(200.dp)
        )

        Spacer(modifier = Modifier.height(16.dp))

        Text(
            text = description,
            fontSize = 16.sp
        )
    }
}
```

HeroViewModel.kt:

Table 27 Source Code Jawaban soal 1 Compose

```
package
com.example.mobilelegendcharacterlist.viewmodel

import androidx.lifecycle.ViewModel
import
com.example.mobilelegendcharacterlist.model.DataHe
ro
import
com.example.mobilelegendcharacterlist.data.HeroDat
a
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.asStateFlow
import timber.log.Timber

class HeroViewModel : ViewModel() {

    private         val         _heroList         =
MutableStateFlow<List<DataHero>>(emptyList())
    val   heroList:   StateFlow<List<DataHero>>   =
_heroList.asStateFlow()


    init {
        loadHeroes()
    }

    private fun loadHeroes() {
        Timber.d("Memuat data hero...")
```

```
30          _heroList.value = HeroData.heroList
31      }
32
33      fun logHeroClick(name: String) {
34          Timber.d("Tombol   'Detail'   diklik   untuk
35  hero: $name")
36      }
37  }
38
39
```

HeroViewModelFactory.kt:

Table 28 Source Code Jawaban soal 1 Compose

```
1   package
2   com.example.mobilelegendcharacterlist.viewmodel
3
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6
7   class HeroViewModelFactory(
8       private val title: String
9   ) : ViewModelProvider.Factory {
10
11      override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
12          if
13  (modelClass.isAssignableFrom(HeroViewModel::class.
    java)) {
14
```

```
15              return HeroViewModel() as T // ganti
jika constructor butuh parameter
16
17          }
18          throw     IllegalArgumentException("Unknown
ViewModel class")
19      }
20  }
```

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

Aplikasi harus dapat mempertahankan fitur-fitur yang sudah dibuat pada modul sebelumnya. Berikut adalah contoh debugging dalam Android Studio.

Dalam arsitektur aplikasi Android, Application class adalah komponen penting yang digunakan untuk mengelola state global dari seluruh aplikasi. Kelas ini merupakan titik awal yang pertama kali diinisialisasi oleh sistem Android sebelum Activity, Service, atau komponen lainnya dijalankan. Fungsinya sangat beragam, mulai dari inisialisasi library pihak ketiga seperti Timber, Firebase, atau Dagger, hingga menyediakan context aplikasi yang bersifat global dan dapat digunakan sepanjang siklus hidup aplikasi. Context ini sangat berguna dalam berbagai kebutuhan, misalnya saat ViewModel atau komponen lain membutuhkan akses ke sumber daya aplikasi tanpa harus bergantung pada context milik Activity.
Selain itu, Application class juga sering dimanfaatkan untuk mencatat peristiwa global seperti logging dan analitik, serta mendukung konsistensi arsitektur, terutama dalam pola MVVM atau Clean Architecture. Sebagai contoh pada praktikum Modul 4, penggunaan Timber untuk logging diinisialisasi di dalam class turunan dari Application, sehingga fitur pencatatan log dapat berjalan di seluruh bagian aplikasi, baik dalam antarmuka berbasis XML maupun Jetpack Compose.
Dengan mendefinisikan Application class dan mendaftarkannya di dalam AndroidManifest.xml, aplikasi dapat mempertahankan dan menjalankan berbagai fitur penting yang telah dibuat pada modul sebelumnya, seperti logging interaksi pengguna, pencatatan event pada ViewModel, dan menjaga efisiensi manajemen dependensi. Hal ini menjadikan Application class sebagai fondasi penting dalam

pengembangan aplikasi Android yang terstruktur, modular, dan mudah dikembangkan lebih lanjut.

MODUL 5: CONNECT TO THE INTERNET

SOAL 1

Soal Praktikum:
Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:

a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
b. Gunakan KotlinX Serialization sebagai library JSON.
c. Gunakan library seperti Coil atau Glide untuk image loading.
d. API yang digunakan pada modul ini adalah The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: https://developer.themoviedb.org/docs/getting-started
e. Implementasikan konsep data persistence (aplikasi menyimpan data walau pengguna keluar dari aplikasi) dengan SharedPreferences untuk menyimpan data ringan (seperti pengaturan aplikasi) dan Room untuk data relasional.
f. Gunakan caching strategy pada Room. Dibebaskan untuk memilih caching strategy yang sesuai, dan sertakan penjelasan kenapa menggunakan caching strategy tersebut.
g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

Aplikasi harus mempertahankan fitur-fitur yang dibuat pada modul sebelumnya.

A. Source Code XML

MainActivity.kt

Table 29 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.presentation.ui.activity
2
3   import android.content.Intent
4   import android.os.Bundle
5   import android.view.View
6   import android.widget.Toast
7   import androidx.activity.viewModels
8   import androidx.appcompat.app.AppCompatActivity
9   import androidx.appcompat.app.AppCompatDelegate
10  import androidx.lifecycle.Observer
11  import androidx.recyclerview.widget.LinearLayoutManager
12  import androidx.room.Room
13  import
14  com.example.movielist.data.local.MovieAppPreferences
15  import
16  com.example.movielist.data.local.database.AppDatabase
17  import
18  com.example.movielist.data.remote.api.RetrofitClient
19  import
20  com.example.movielist.data.repository.MovieRepositoryImp
21  l
22  import
23  com.example.movielist.databinding.ActivityMainBinding
24  import
25  com.example.movielist.domain.usecase.GetPopularMoviesUse
26  Case
27  import
28  com.example.movielist.presentation.ui.adapter.MovieAdapt
29  er
30  import
31  com.example.movielist.presentation.viewmodel.MovieViewMo
32  del
33  import
34  com.example.movielist.presentation.viewmodel.ViewModelFa
35  ctory
36  import com.example.movielist.utils.Result
37
38  class MainActivity : AppCompatActivity() {
39
40      private lateinit var binding: ActivityMainBinding
```

```
41      private lateinit var movieAdapter: MovieAdapter
42      private lateinit var movieAppPreferences:
43  MovieAppPreferences
44
45      private val movieViewModel: MovieViewModel by
46  viewModels {
47          val apiService = RetrofitClient.tmdbApiService
48          val database = Room.databaseBuilder(
49              applicationContext,
50              AppDatabase::class.java,
51              AppDatabase.DATABASE_NAME
52          ).build()
53          val movieDao = database.movieDao()
54
55          val tmdbApiKey =
56  "71819bfeac768c2a5b9a32b26e50cae1"
57          movieAppPreferences.saveApiKey(tmdbApiKey)
58
59          val movieRepositoryImpl =
60  MovieRepositoryImpl(apiService, movieDao, tmdbApiKey)
61          val getPopularMoviesUseCase =
62  GetPopularMoviesUseCase(movieRepositoryImpl)
63          ViewModelFactory(getPopularMoviesUseCase)
64      }
65
66      override fun onCreate(savedInstanceState: Bundle?) {
67          super.onCreate(savedInstanceState)
68          binding =
69  ActivityMainBinding.inflate(layoutInflater)
70          setContentView(binding.root)
71
72          movieAppPreferences = MovieAppPreferences(this)
73
74          setupRecyclerView()
75          observeViewModel()
76          setupDarkModeToggle()
77
78          binding.btnRetry.setOnClickListener {
79              movieViewModel.fetchPopularMovies()
80          }
81      }
82
83      private fun setupRecyclerView() {
84          movieAdapter = MovieAdapter()
85          binding.rvMovies.apply {
```

```
86              layoutManager =
87  LinearLayoutManager(this@MainActivity)
88              adapter = movieAdapter
89          }
90
91          movieAdapter.onItemClick = { movie ->
92              val intent = Intent(this,
93  DetailActivity::class.java).apply {
94                  putExtra(DetailActivity.EXTRA_MOVIE,
95  movie)
96              }
97              startActivity(intent)
98          }
99      }
100
101     private fun observeViewModel() {
102         movieViewModel.popularMovies.observe(this,
103 Observer { result ->
104             when (result) {
105                 is Result.Loading -> {
106                     binding.progressBar.visibility =
107 View.VISIBLE
108                     binding.tvError.visibility =
109 View.GONE
110                     binding.btnRetry.visibility =
111 View.GONE
112                     binding.rvMovies.visibility =
113 View.GONE
114                 }
115                 is Result.Success -> {
116                     binding.progressBar.visibility =
117 View.GONE
118                     binding.tvError.visibility =
119 View.GONE
120                     binding.btnRetry.visibility =
121 View.GONE
122                     binding.rvMovies.visibility =
123 View.VISIBLE
124                     movieAdapter.submitList(result.data)
125                 }
126                 is Result.Error -> {
127                     binding.progressBar.visibility =
128 View.GONE
129                     binding.rvMovies.visibility =
130 View.GONE
```

```
                        binding.tvError.visibility =
View.VISIBLE
                        binding.btnRetry.visibility =
View.VISIBLE
                        binding.tvError.text = "Error:
${result.exception.message}"
                        Toast.makeText(this, "Error:
${result.exception.message}", Toast.LENGTH_LONG).show()
                    }
                }
            })
        }

        private fun setupDarkModeToggle() {
            binding.switchDarkMode.isChecked =
movieAppPreferences.getDarkModeState()


applyTheme(movieAppPreferences.getDarkModeState())


binding.switchDarkMode.setOnCheckedChangeListener { _,
isChecked ->

movieAppPreferences.saveDarkModeState(isChecked)
                applyTheme(isChecked)
            }
        }

        private fun applyTheme(isDarkMode: Boolean) {
            if (isDarkMode) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.
MODE_NIGHT_YES)
            } else {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.
MODE_NIGHT_NO)
            }
        }
}
```

MovieDao.kt

Table 30 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.data.local.dao
2
3   import androidx.room.Dao
4   import androidx.room.Insert
5   import androidx.room.OnConflictStrategy
6   import androidx.room.Query
7   import
8   com.example.movielist.data.local.entities.MovieEntity
9
10  @Dao
11  interface MovieDao {
12      @Insert(onConflict = OnConflictStrategy.REPLACE)
13      suspend fun insertAllMovies(movies:
14  List<MovieEntity>)
15
16      @Query("SELECT * FROM movies ORDER BY popularity
17  DESC")
18      suspend fun getAllMovies(): List<MovieEntity>
19
        @Query("DELETE FROM movies")
        suspend fun clearAllMovies()
    }
```

AppDatabase.kt

Table 31 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.data.local.database
2
3   import androidx.room.Database
4   import androidx.room.RoomDatabase
5   import com.example.movielist.data.local.dao.MovieDao
6   import
7   com.example.movielist.data.local.entities.MovieEntity
8
9   @Database(entities = [MovieEntity::class], version = 1,
10  exportSchema = false)
11  abstract class AppDatabase : RoomDatabase() {
12      abstract fun movieDao(): MovieDao
```

```
13
14      companion object {
15          const val DATABASE_NAME = "tmdb_app_db"
16      }
    }
```

MovieEntity.kt

Table 32 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.data.local.entities
2
3   import androidx.room.Entity
4   import androidx.room.PrimaryKey
5   import com.example.movielist.domain.model.Movie
6
7   @Entity(tableName = "movies")
8   data class MovieEntity(
9       @PrimaryKey
10      val id: Int,
11      val title: String,
12      val overview: String,
13      val posterPath: String?,
14      val releaseDate: String,
15      val voteAverage: Double,
16      val popularity: Double
17  ) {
18      fun toDomainMovie(): Movie {
19          return Movie(
20              id = id,
21              title = title,
22              overview = overview,
23              posterPath = posterPath,
24              releaseDate = releaseDate,
25              voteAverage = voteAverage
26          )
27      }
28
29      companion object {
30          fun fromDomainMovie(movie: Movie, popularity:
31  Double): MovieEntity {
32              return MovieEntity(
33                  id = movie.id,
```

```
                    title = movie.title,
                    overview = movie.overview,
                    posterPath = movie.posterPath,
                    releaseDate = movie.releaseDate,
                    voteAverage = movie.voteAverage,
                    popularity = popularity
                )
            }
        }
}
```

MovieAppPreferences.kt

Table 33 Source Code Jawaban soal 1 XML

```
package com.example.movielist.data.local

import android.content.Context
import android.content.SharedPreferences

class MovieAppPreferences(context: Context) {

    private val sharedPreferences: SharedPreferences =
        context.getSharedPreferences("tmdb_app_prefs",
Context.MODE_PRIVATE)

    companion object {
        private const val KEY_API_KEY = "api_key"
        private const val KEY_DARK_MODE = "dark_mode"
    }

    fun saveApiKey(apiKey: String) {
        sharedPreferences.edit().putString(KEY_API_KEY,
apiKey).apply()
    }

    fun getApiKey(): String? {
        return sharedPreferences.getString(KEY_API_KEY,
null)
    }

```

```
27      fun saveDarkModeState(isDarkMode: Boolean) {
28
29  sharedPreferences.edit().putBoolean(KEY_DARK_MODE,
30  isDarkMode).apply()
31      }
32
33      fun getDarkModeState(): Boolean {
34          return
    sharedPreferences.getBoolean(KEY_DARK_MODE, false)
        }
    }
```

RetrofitClient.kt

Table 34 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.data.remote.api
2
3   import
4   com.jakewharton.retrofit2.converter.kotlinx.serializatio
5   n.asConverterFactory
6   import kotlinx.serialization.json.Json
7   import okhttp3.MediaType.Companion.toMediaType
8   import okhttp3.OkHttpClient
9   import okhttp3.logging.HttpLoggingInterceptor
10  import retrofit2.Retrofit
11  import java.util.concurrent.TimeUnit
12
13  object RetrofitClient {
14
15      private const val BASE_URL =
16  "https://api.themoviedb.org/3/"
17
18      private val json = Json {
19          ignoreUnknownKeys = true
20          prettyPrint = true
21      }
22
23      private val okHttpClient: OkHttpClient by lazy {
24          val logging = HttpLoggingInterceptor()
25
```

```
logging.setLevel(HttpLoggingInterceptor.Level.BODY)

        OkHttpClient.Builder()
            .addInterceptor(logging)
            .connectTimeout(30, TimeUnit.SECONDS)
            .readTimeout(30, TimeUnit.SECONDS)
            .writeTimeout(30, TimeUnit.SECONDS)
            .build()
    }

    val tmdbApiService: TmdbApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .client(okHttpClient)

.addConverterFactory(json.asConverterFactory("applicatio
n/json".toMediaType()))
            .build()
            .create(TmdbApiService::class.java)
    }
}
```

***TmdbApiService*.kt**

Table 35 Source Code Jawaban soal 1 XML

```
package com.example.movielist.data.remote.api

import
com.example.movielist.data.remote.models.MovieListRespon
se
import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Query

interface TmdbApiService {

    @GET("movie/popular")
    suspend fun getPopularMovies(
        @Query("api_key") apiKey: String,
```

```
        @Query("language") language: String = "en-US",
        @Query("page") page: Int = 1
    ): Response<MovieListResponse>
}
```

MovieDto.kt

Table 36 Source Code Jawaban soal 1 XML

```
package com.example.movielist.data.remote.models

import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
data class MovieDto(
    val adult: Boolean,
    @SerialName("backdrop_path")
    val backdropPath: String?,
    @SerialName("genre_ids")
    val genreIds: List<Int>,
    val id: Int,
    @SerialName("original_language")
    val originalLanguage: String,
    @SerialName("original_title")
    val originalTitle: String,
    val overview: String,
    val popularity: Double,
    @SerialName("poster_path")
    val posterPath: String?,
    @SerialName("release_date")
    val releaseDate: String,
    val title: String,
    val video: Boolean,
    @SerialName("vote_average")
    val voteAverage: Double,
    @SerialName("vote_count")
    val voteCount: Int
)
```

MovieDtoExtension.kt

Table 37 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.data.remote.models
2
3   import com.example.movielist.domain.model.Movie
4   import
5   com.example.movielist.data.local.entities.MovieEntity
6
7   fun MovieDto.toDomainMovie(): Movie {
8       return Movie(
9           id = id,
10          title = title,
11          overview = overview,
12          posterPath = posterPath,
13          releaseDate = releaseDate,
14          voteAverage = voteAverage
15      )
16  }
17
18  fun MovieDto.toMovieEntity(): MovieEntity {
19      return MovieEntity(
20          id = id,
21          title = title,
22          overview = overview,
23          posterPath = posterPath,
24          releaseDate = releaseDate,
25          voteAverage = voteAverage,
26          popularity = popularity
27      )
    }
```

MovieListResponse.kt

Table 38 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.data.remote.models
2
3   import kotlinx.serialization.SerialName
4   import kotlinx.serialization.Serializable
5
```

```
@Serializable
data class MovieListResponse(
    val page: Int,
    val results: List<MovieDto>,
    @SerialName("total_pages")
    val totalPages: Int,
    @SerialName("total_results")
    val totalResults: Int
)
```

MovieRepository.kt

Table 39 Source Code Jawaban soal 1 XML

```
package com.example.movielist.data.repository

import com.example.movielist.data.local.dao.MovieDao
import
com.example.movielist.data.remote.api.TmdbApiService
import
com.example.movielist.data.remote.models.toDomainMovie
import
com.example.movielist.data.remote.models.toMovieEntity
import com.example.movielist.domain.model.Movie
import com.example.movielist.utils.Result
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow
import retrofit2.HttpException
import java.io.IOException

interface MovieRepository {
    fun getPopularMovies(): Flow<Result<List<Movie>>>
}

class MovieRepositoryImpl(
    private val apiService: TmdbApiService,
    private val movieDao: MovieDao,
    private val apiKey: String
) : MovieRepository {

    override fun getPopularMovies():
Flow<Result<List<Movie>>> = flow {
        emit(Result.Loading)
```

```
30
31          val cachedMovies = movieDao.getAllMovies().map
32  { it.toDomainMovie() }
33          if (cachedMovies.isNotEmpty()) {
34              emit(Result.Success(cachedMovies))
35          }
36
37          try {
38              val response =
39  apiService.getPopularMovies(apiKey = apiKey)
40              if (response.isSuccessful) {
41                  val movieDtos =
42  response.body()?.results ?: emptyList()
43                  val domainMovies = movieDtos.map {
44  it.toDomainMovie() }
45
46                  movieDao.clearAllMovies()
47                  movieDao.insertAllMovies(movieDtos.map
48  { it.toMovieEntity() })
49
50                  emit(Result.Success(domainMovies))
51              } else {
52                  emit(Result.Error(Exception("API Error:
53  ${response.code()} ${response.message()}")))
54              }
55          } catch (e: HttpException) {
56              emit(Result.Error(Exception("Network Error
57  (HTTP ${e.code()}): ${e.message()}")))
58          } catch (e: IOException) {
59              emit(Result.Error(Exception("No Internet
60  Connection or API Timeout: ${e.message}")))
61          } catch (e: Exception) {
62              emit(Result.Error(Exception("An unexpected
63  error occurred: ${e.localizedMessage}")))
            }
      }
  }
```

Movie.kt

Table 40 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.domain.model
2
3   import android.os.Parcelable
4   import kotlinx.parcelize.Parcelize
5
6   @Parcelize
7   data class Movie(
8       val id: Int,
9       val title: String,
10      val overview: String,
11      val posterPath: String?,
12      val releaseDate: String,
13      val voteAverage: Double
14  ) : Parcelable
```

GetPopularMoviesUseCase.kt

Table 41 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.domain.usecase
2
3   import com.example.movielist.domain.model.Movie
4   import
5   com.example.movielist.data.repository.MovieRepositoryI
6   mpl
7   import com.example.movielist.utils.Result
8   import kotlinx.coroutines.flow.Flow
9
10  class GetPopularMoviesUseCase (
11      private val movieRepository: MovieRepositoryImpl
12  ) {
13      operator fun invoke(): Flow<Result<List<Movie>>> {
14          return movieRepository.getPopularMovies()
        }
    }
```

DetailActivity.kt

Table 42 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.presentation.ui.activity
2
3   import android.os.Build
4   import android.os.Bundle
5   import android.view.MenuItem
6   import android.widget.Toast
7   import androidx.appcompat.app.AppCompatActivity
8   import com.bumptech.glide.Glide
9   import
10  com.example.movielist.databinding.ActivityDetailBindin
11  g
12  import com.example.movielist.domain.model.Movie
13
14  class DetailActivity : AppCompatActivity() {
15
16      private lateinit var binding:
17  ActivityDetailBinding
18
19      companion object {
20          const val EXTRA_MOVIE = "extra_movie"
21      }
22
23      override fun onCreate(savedInstanceState: Bundle?)
24  {
25          super.onCreate(savedInstanceState)
26          binding =
27  ActivityDetailBinding.inflate(layoutInflater)
28          setContentView(binding.root)
29
30
31
32  supportActionBar?.setDisplayHomeAsUpEnabled(true)
33
34
35          val movie = if (Build.VERSION.SDK_INT >=
36  Build.VERSION_CODES.TIRAMISU) {
37              intent.getParcelableExtra(EXTRA_MOVIE,
38  Movie::class.java)
39          } else {
40              @Suppress("DEPRECATION")
41              intent.getParcelableExtra(EXTRA_MOVIE)
```

```
        }

        movie?.let {

            supportActionBar?.title = it.title

            binding.apply {
                tvDetailTitle.text = it.title
                tvDetailReleaseDate.text = "Release
Date: ${it.releaseDate}"
                tvDetailVoteAverage.text = "Rating:
${String.format("%.1f", it.voteAverage)}"
                tvDetailOverview.text = it.overview

                val imageUrl =
"https://image.tmdb.org/t/p/w500${it.posterPath}"
                Glide.with(this@DetailActivity)
                    .load(imageUrl)
                    .centerCrop()
                    .into(ivDetailPoster)
            }
        } ?: run {
            Toast.makeText(this, "Film tidak
ditemukan.", Toast.LENGTH_SHORT).show()
            finish()
        }
    }


    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        if (item.itemId == android.R.id.home) {
            onBackPressedDispatcher.onBackPressed()
            return true
        }
        return super.onOptionsItemSelected(item)
    }
}
```

DetailActivity.kt

Table 43 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.presentation.ui.adapter
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.DiffUtil
6   import androidx.recyclerview.widget.ListAdapter
7   import androidx.recyclerview.widget.RecyclerView
8   import com.bumptech.glide.Glide
9   import
10  com.example.movielist.databinding.ItemMovieBinding
11  import com.example.movielist.domain.model.Movie
12
13  class MovieAdapter : ListAdapter<Movie,
14  MovieAdapter.MovieViewHolder>(MovieDiffCallback()) {
15
16      var onItemClick: ((Movie) -> Unit)? = null
17
18      override fun onCreateViewHolder(parent: ViewGroup,
19  viewType: Int): MovieViewHolder {
20          val binding =
21  ItemMovieBinding.inflate(LayoutInflater.from(parent.con
22  text), parent, false)
23          return MovieViewHolder(binding)
24      }
25
26      override fun onBindViewHolder(holder:
27  MovieViewHolder, position: Int) {
28          val movie = getItem(position)
29          holder.bind(movie)
30      }
31
32      inner class MovieViewHolder(private val binding:
33  ItemMovieBinding) :
34          RecyclerView.ViewHolder(binding.root) {
35
36          init {
37              binding.btnDetail.setOnClickListener {
38
39  onItemClick?.invoke(getItem(adapterPosition))
40              }
41          }
```

```
42
43          fun bind(movie: Movie) {
44              binding.apply {
45                  tvMovieTitle.text = movie.title
46                  tvReleaseDate.text = "Release Date:
47  ${movie.releaseDate}"
48                  tvVoteAverage.text = "Rating:
49  ${String.format("%.1f", movie.voteAverage)}"
50                  tvOverview.text = movie.overview
51
52                  val imageUrl =
53  "https://image.tmdb.org/t/p/w500${movie.posterPath}"
54
55                  Glide.with(itemView.context)
56                      .load(imageUrl)
57                      .centerCrop()
58                      .into(ivPoster)
59              }
60          }
61      }
62
63      class MovieDiffCallback :
64  DiffUtil.ItemCallback<Movie>() {
65          override fun areItemsTheSame(oldItem: Movie,
66  newItem: Movie): Boolean {
67              return oldItem.id == newItem.id
68          }
69
70          override fun areContentsTheSame(oldItem: Movie,
71  newItem: Movie): Boolean {
72              return oldItem == newItem
            }
        }
    }
```

MovieViewModel.kt

Table 44 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.presentation.viewmodel
2
3   import androidx.lifecycle.LiveData
4   import androidx.lifecycle.MutableLiveData
```

```
5   import androidx.lifecycle.ViewModel
6   import androidx.lifecycle.viewModelScope
7   import com.example.movielist.domain.model.Movie
8   import
9   com.example.movielist.domain.usecase.GetPopularMoviesUs
10  eCase
11  import com.example.movielist.utils.Result
12  import kotlinx.coroutines.launch
13
14  class MovieViewModel (
15      private val getPopularMoviesUseCase:
16  GetPopularMoviesUseCase
17  ) : ViewModel() {
18
19      private val _popularMovies =
20  MutableLiveData<Result<List<Movie>>>()
21      val popularMovies: LiveData<Result<List<Movie>>> =
22  _popularMovies
23
24      init {
25          fetchPopularMovies()
26      }
27
28      fun fetchPopularMovies() {
29          viewModelScope.launch {
30              getPopularMoviesUseCase().collect { result
31  ->
                    _popularMovies.value = result
                }
            }
        }
    }
```

ViewModelFactory.kt

Table 45 Source Code Jawaban soal 1 XML

```
1   package com.example.movielist.presentation.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5   import
6   com.example.movielist.domain.usecase.GetPopularMoviesUs
```

```
eCase

class ViewModelFactory(
    private val getPopularMoviesUseCase:
GetPopularMoviesUseCase
) : ViewModelProvider.Factory {

    override fun <T : ViewModel> create(modelClass:
Class<T>): T {
        if
(modelClass.isAssignableFrom(MovieViewModel::class.java
)) {
            @Suppress("UNCHECKED_CAST")
            return
MovieViewModel(getPopularMoviesUseCase) as T
        }
        throw IllegalArgumentException("Unknown
ViewModel class")
    }
}
```

Result.kt

Table 46 Source Code Jawaban soal 1 XML

```
package com.example.movielist.utils

sealed class Result<out T> {
    object Loading : Result<Nothing>()
    data class Success<out T>(val data: T) :
Result<T>()
    data class Error(val exception: Exception) :
Result<Nothing>()
}
```

Dalam file layout:

activity_main.xml

Dalam XML ada beberapa file tambahan agar sama tampilannya dengan di gambar.

Table 47 Source Code Jawaban soal 1 XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/androi
4   d"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:layout_width="match_parent"
8       android:layout_height="match_parent"
9
10  tools:context=".presentation.ui.activity.MainActivity">
11
12      <TextView
13          android:id="@+id/tv_title"
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content"
16          android:layout_marginTop="8dp"
17          android:text="Popular Movies"
18          android:textSize="24sp"
19          android:textStyle="bold"
20          app:layout_constraintHorizontal_bias="0.454"
21          app:layout_constraintStart_toStartOf="parent"
22          app:layout_constraintTop_toTopOf="parent" />
23
24      <ProgressBar
25          android:id="@+id/progress_bar"
26          android:layout_width="wrap_content"
27          android:layout_height="wrap_content"
28          android:visibility="gone"
29
30  app:layout_constraintTop_toBottomOf="@id/tv_title"
31          app:layout_constraintStart_toStartOf="parent"
32          app:layout_constraintEnd_toEndOf="parent"
33          app:layout_constraintBottom_toBottomOf="parent"
34  />
35
36      <TextView
37          android:id="@+id/tv_error"
38          android:layout_width="wrap_content"
39          android:layout_height="wrap_content"
40          android:text="Error: Could not load movies."
41          android:textColor="@android:color/holo_red_dark"
```

```
        android:visibility="gone"
        android:gravity="center"

app:layout_constraintTop_toBottomOf="@id/tv_title"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
/>

    <Button
        android:id="@+id/btn_retry"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Retry"
        android:visibility="gone"
        android:layout_marginTop="8dp"

app:layout_constraintTop_toBottomOf="@id/tv_error"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
/>


    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_movies"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginTop="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/tv_title"
        tools:listitem="@layout/item_movie" />


<com.google.android.material.switchmaterial.SwitchMateri
al
        android:id="@+id/switch_dark_mode"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="Dark Mode"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

item_movie.xml:

Table 48 Source Code Jawaban soal 1 XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
3   xmlns:android="http://schemas.android.com/apk/res/androi
4   d"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:layout_width="match_parent"
8       android:layout_height="wrap_content"
9       android:layout_margin="8dp"
10      app:cardCornerRadius="8dp"
11      app:cardElevation="4dp">
12
13      <androidx.constraintlayout.widget.ConstraintLayout
14          android:layout_width="match_parent"
15          android:layout_height="wrap_content"
16          android:padding="16dp">
17
18          <ImageView
19              android:id="@+id/iv_poster"
20              android:layout_width="100dp"
21              android:layout_height="150dp"
22              android:scaleType="centerCrop"
23
24  app:layout_constraintStart_toStartOf="parent"
25              app:layout_constraintTop_toTopOf="parent"
26              tools:src="@tools:sample/avatars" />
27
28          <TextView
29              android:id="@+id/tv_movie_title"
30              android:layout_width="0dp"
31              android:layout_height="wrap_content"
32              android:layout_marginStart="16dp"
33              android:textStyle="bold"
```

```
34              android:textSize="18sp"
35              app:layout_constraintEnd_toEndOf="parent"
36
37  app:layout_constraintStart_toEndOf="@id/iv_poster"
38
39  app:layout_constraintTop_toTopOf="@id/iv_poster"
40              tools:text="Movie Title" />
41
42          <TextView
43              android:id="@+id/tv_release_date"
44              android:layout_width="0dp"
45              android:layout_height="wrap_content"
46              android:layout_marginStart="16dp"
47              android:layout_marginTop="4dp"
48              android:textSize="14sp"
49              app:layout_constraintEnd_toEndOf="parent"
50
51  app:layout_constraintStart_toEndOf="@id/iv_poster"
52
53  app:layout_constraintTop_toBottomOf="@id/tv_movie_title"
54              tools:text="Release Date: 2023-01-01" />
55
56          <TextView
57              android:id="@+id/tv_vote_average"
58              android:layout_width="0dp"
59              android:layout_height="wrap_content"
60              android:layout_marginStart="16dp"
61              android:layout_marginTop="4dp"
62              android:textSize="14sp"
63              app:layout_constraintEnd_toEndOf="parent"
64
65  app:layout_constraintStart_toEndOf="@id/iv_poster"
66
67  app:layout_constraintTop_toBottomOf="@id/tv_release_date
68  "
69              tools:text="Rating: 7.5" />
70
71          <TextView
72              android:id="@+id/tv_overview"
73              android:layout_width="0dp"
74              android:layout_height="wrap_content"
75              android:layout_marginTop="8dp"
76              android:maxLines="3"
77              android:ellipsize="end"
78              app:layout_constraintEnd_toEndOf="parent"
```

```
79
80  app:layout_constraintStart_toStartOf="parent"
81
82  app:layout_constraintTop_toBottomOf="@id/iv_poster"
83              tools:text="This is a short overview of the
84  movie. It talks about the plot and characters..." />
85
            <Button
                android:id="@+id/btn_detail"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="16dp"
                android:text="Detail"
                app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintTop_toBottomOf="@id/tv_overview" />

        </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
```

activity_detail.xml

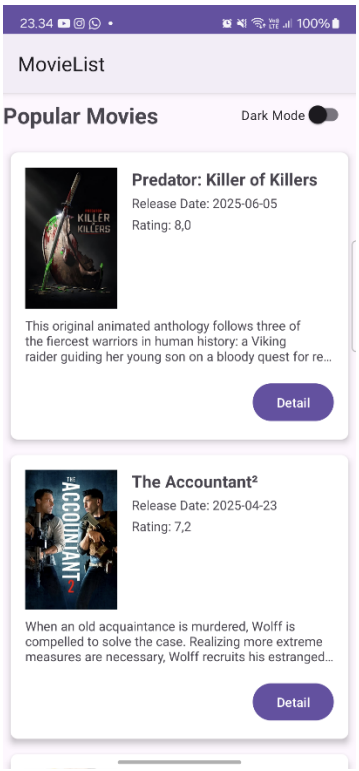
Table 49 Source Code Jawaban soal 1 XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <ScrollView
3   xmlns:android="http://schemas.android.com/apk/res/androi
4   d"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:layout_width="match_parent"
8       android:layout_height="match_parent"
9
10  tools:context=".presentation.ui.activity.DetailActivity"
11  >
12
13      <androidx.constraintlayout.widget.ConstraintLayout
14          android:layout_width="match_parent"
15          android:layout_height="wrap_content"
16          android:padding="16dp">
17
18          <ImageView
19              android:id="@+id/iv_detail_poster"
20              android:layout_width="0dp"
21              android:layout_height="300dp"
```

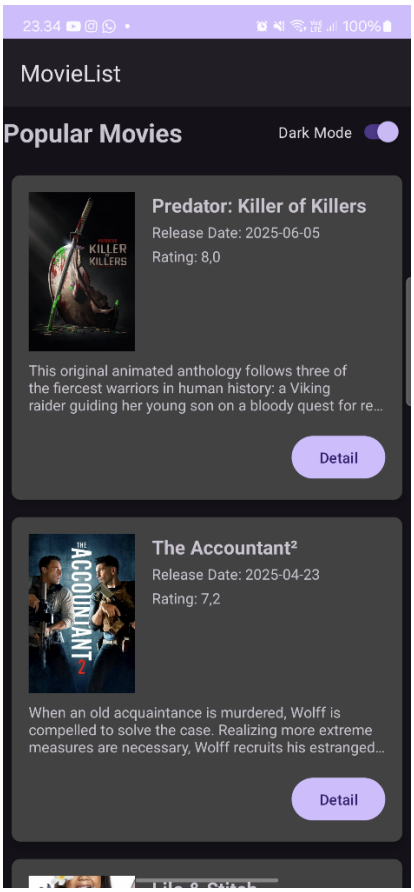
```
            android:scaleType="centerCrop"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:src="@tools:sample/avatars" />

        <TextView
            android:id="@+id/tv_detail_title"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:textSize="24sp"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/iv_detail_poste
r"
            tools:text="Movie Title on Detail Page" />

        <TextView
            android:id="@+id/tv_detail_release_date"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:textSize="16sp"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/tv_detail_title
"
            tools:text="Release Date: 2023-01-01" />

        <TextView
            android:id="@+id/tv_detail_vote_average"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
            android:textSize="16sp"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"
```

```
67
68  app:layout_constraintTop_toBottomOf="@id/tv_detail_relea
69  se_date"
70              tools:text="Rating: 7.5" />
71
72          <TextView
73              android:id="@+id/tv_detail_overview_label"
74              android:layout_width="0dp"
75              android:layout_height="wrap_content"
76              android:layout_marginTop="16dp"
77              android:text="Overview:"
78              android:textSize="18sp"
79              android:textStyle="bold"
80              app:layout_constraintEnd_toEndOf="parent"
81
82  app:layout_constraintStart_toStartOf="parent"
83
84  app:layout_constraintTop_toBottomOf="@id/tv_detail_vote_
85  average" />
86
87          <TextView
88              android:id="@+id/tv_detail_overview"
89              android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:textSize="16sp"
                app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@id/tv_detail_overv
    iew_label"
                tools:text="This is a very long and detailed
    overview of the movie. It covers the plot, characters,
    themes, and critical reception." />

        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>
```

B. Output Program

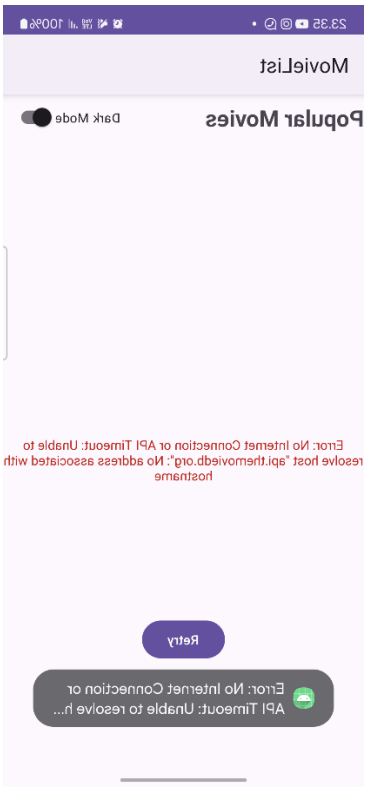


Gambar 27. Screenshot Hasil Jawaban Soal 1 XML



Gambar 28. Screenshot Hasil Jawaban Soal 1 XML(darkmode)

Gambar 29. Detail Button



Gambar 30. Error Button

C. Pembahasan

MainActivity.kt:

Pada baris 1, package com.example.movielist.presentation.ui.activity mendefinisikan package untuk Activity utama, bagian dari lapisan presentasi.

Pada baris 3-21, import berbagai class dan interface yang dibutuhkan untuk Activity (Intent, Bundle, View, Toast, ViewModel, dll.).

Pada baris 23, class MainActivity : AppCompatActivity() mendefinisikan class MainActivity yang merupakan turunan dari AppCompatActivity.

Pada baris 26, private lateinit var binding: ActivityMainBinding mendeklarasikan variabel binding untuk View Binding, yang memungkinkan akses mudah ke elemen UI di activity_main.xml. lateinit berarti akan diinisialisasi nanti.

Pada baris 27, private lateinit var movieAdapter: MovieAdapter mendeklarasikan adapter untuk RecyclerView yang akan menampilkan daftar film.

Pada baris 28, private lateinit var movieAppPreferences: MovieAppPreferences mendeklarasikan instance MovieAppPreferences untuk mengelola SharedPreferences.

Pada baris 30, private var darkModeSwitch: SwitchMaterial? = null mendeklarasikan variabel darkModeSwitch yang akan menampung referensi ke SwitchMaterial di menu.

Pada baris 32, private val movieViewModel: MovieViewModel by viewModels { ... } mendeklarasikan dan menginisialisasi MovieViewModel menggunakan by viewModels delegate. by viewModels memastikan ViewModel bertahan hidup saat konfigurasi berubah. Blok { ... } adalah tempat Dependensi diinjeksi secara manual untuk ViewModelFactory.

Pada baris 33-41, Inisialisasi dependensi: apiService dari RetrofitClient, database Room, movieDao, tmdbApiKey (disimpan ke preferences), movieRepositoryImpl, dan getPopularMoviesUseCase. Semua ini kemudian dilewatkan ke ViewModelFactory.

Pada baris 44, override fun onCreate(savedInstanceState: Bundle?) adalah method yang dipanggil saat Activity pertama kali dibuat.

Pada baris 45, super.onCreate(savedInstanceState) memanggil implementasi onCreate dari superclass (AppCompatActivity).

Pada baris 46, binding = ActivityMainBinding.inflate(layoutInflater) menginisialisasi View Binding.

Pada baris 47, setContentView(binding.root) menetapkan root layout dari binding sebagai tampilan Activity.

Pada baris 49, supportActionBar?.setDisplayHomeAsUpEnabled(false) menonaktifkan tombol back di ActionBar untuk MainActivity.

Pada baris 50, supportActionBar?.title = "Popular Movies" mengatur judul ActionBar menjadi "Popular Movies".

Pada baris 52, movieAppPreferences = MovieAppPreferences(this) menginisialisasi MovieAppPreferences.

Pada baris 54-56, setupRecyclerView(), observeViewModel(), dan setupDarkModeToggle() dipanggil untuk menyiapkan UI, mengamati data, dan mengelola mode gelap.

Pada baris 58, binding.btnRetry.setOnClickListener { ... } mengatur listener klik untuk tombol "Retry".

Pada baris 59, movieViewModel.fetchPopularMovies() memanggil fungsi di ViewModel untuk memuat ulang data saat tombol "Retry" diklik.

Pada baris 63, override fun onCreateOptionsMenu(menu: Menu?): Boolean adalah method yang dipanggil untuk membuat menu opsi di ActionBar.

Pada baris 64, menuInflater.inflate(R.menu.main_menu, menu) meng-inflate layout menu XML (main_menu.xml) ke dalam ActionBar.

Pada baris 66, val darkModeMenuItem = menu?.findItem(R.id.action_dark_mode_toggle) menemukan MenuItem untuk dark mode berdasarkan ID-nya.

Pada baris 67, darkModeSwitch = darkModeMenuItem?.actionView as? SwitchMaterial mendapatkan referensi ke SwitchMaterial yang merupakan actionLayout dari MenuItem.

Pada baris 69, darkModeSwitch?.apply { ... } adalah blok apply untuk mengkonfigurasi SwitchMaterial jika tidak null.

Pada baris 70, isChecked = movieAppPreferences.getDarkModeState() mengatur status awal SwitchMaterial berdasarkan preferensi yang tersimpan.

Pada baris 71, applyTheme(isChecked) menerapkan tema (dark/light) saat aplikasi dimulai.

Pada baris 73, setOnCheckedChangeListener { _, isChecked -> ... } mengatur listener untuk perubahan status SwitchMaterial.

Pada baris 74, movieAppPreferences.saveDarkModeState(isChecked) menyimpan status mode gelap yang baru ke SharedPreferences.

Pada baris 75, applyTheme(isChecked) menerapkan tema baru, yang akan membuat ulang Activity.

Pada baris 81, private fun setupRecyclerView() mendefinisikan fungsi untuk menyiapkan RecyclerView.

Pada baris 82, movieAdapter = MovieAdapter() menginisialisasi MovieAdapter.

Pada baris 83-86, binding.rvMovies.apply { ... } mengatur LinearLayoutManager dan adapter untuk RecyclerView.

Pada baris 88, movieAdapter.onItemClick = { movie -> ... } mengatur lambda callback yang akan dipanggil saat tombol "Detail" di item RecyclerView diklik. Sekarang menerima objek Movie.

Pada baris 89, val intent = Intent(this, DetailActivity::class.java) membuat Intent untuk meluncurkan DetailActivity.

Pada baris 90, putExtra(DetailActivity.EXTRA_MOVIE, movie) menambahkan objek Movie sebagai extra ke Intent untuk diteruskan ke DetailActivity.

Pada baris 92, startActivity(intent) meluncurkan DetailActivity.

Pada baris 95, private fun observeViewModel() mendefinisikan fungsi untuk mengamati LiveData dari ViewModel.

Pada baris 96, movieViewModel.popularMovies.observe(this, Observer { result -> ... }) mengamati LiveData popularMovies dari movieViewModel. Blok Observer akan dieksekusi setiap kali nilai LiveData berubah.

Pada baris 97, when (result) { ... } adalah ekspresi when yang memeriksa status Result (Loading, Success, Error) dan memperbarui UI sesuai.

Pada baris 98, is Result.Loading -> { ... } menangani status loading: ProgressBar terlihat, elemen lain disembunyikan.

Pada baris 104, is Result.Success -> { ... } menangani status sukses: ProgressBar dan error/retry disembunyikan, RecyclerView terlihat, dan data disubmit ke adapter.

Pada baris 110, is Result.Error -> { ... } menangani status error: ProgressBar dan RecyclerView disembunyikan, pesan error dan tombol retry terlihat, dan Toast ditampilkan.

Pada baris 119, private fun applyTheme(isDarkMode: Boolean) mendefinisikan fungsi untuk menerapkan tema terang atau gelap.

Pada baris 120-123, AppCompatDelegate.setDefaultNightMode(...) adalah inti dari logika dark mode, yang memberitahu sistem untuk menggunakan mode malam atau tidak. Ini akan menyebabkan Activity dibuat ulang.

MovieDao.kt
Pada baris 1, package com.example.movielist.data.local.dao mendefinisikan package untuk Data Access Object (DAO) film, bagian dari lapisan data lokal.

Pada baris 3, import androidx.room.Dao mengimpor anotasi @Dao, yang menandai antarmuka sebagai DAO Room.

Pada baris 4, import androidx.room.Insert mengimpor anotasi @Insert, digunakan untuk operasi penyisipan data.

Pada baris 5, import androidx.room.OnConflictStrategy mengimpor enum untuk strategi penanganan konflik saat penyisipan.

Pada baris 6, import androidx.room.Query mengimpor anotasi @Query, digunakan untuk kueri SQL kustom.

Pada baris 7, import com.example.movielist.data.local.entities.MovieEntity mengimpor class MovieEntity, entitas Room yang akan dioperasikan.

Pada baris 9, @Dao menandai antarmuka ini sebagai DAO.

Pada baris 10, interface MovieDao mendeklarasikan antarmuka MovieDao.

Pada baris 11, @Insert(onConflict = OnConflictStrategy.REPLACE) menandai fungsi ini sebagai operasi penyisipan. OnConflictStrategy.REPLACE berarti jika ada konflik (misalnya, ID yang sama), data lama akan diganti.

Pada baris 12, suspend fun insertAllMovies(movies: List<MovieEntity>) mendefinisikan fungsi suspend untuk menyisipkan daftar MovieEntity. Kata kunci suspend menunjukkan bahwa ini adalah fungsi coroutine yang dapat dihentikan (pausable) dan dilanjutkan.

Pada baris 14, @Query("SELECT * FROM movies ORDER BY popularity DESC") menandai fungsi ini dengan kueri SQL kustom untuk mengambil semua film dari tabel "movies" dan mengurutkannya berdasarkan popularitas secara descending.

Pada baris 15, suspend fun getAllMovies(): List<MovieEntity> mendefinisikan fungsi suspend untuk mengambil semua MovieEntity.

Pada baris 17, @Query("DELETE FROM movies") menandai fungsi ini dengan kueri SQL kustom untuk menghapus semua data dari tabel "movies".

Pada baris 18, suspend fun clearAllMovies() mendefinisikan fungsi suspend untuk menghapus semua film dari cache.

AppDatabase.kt
Pada baris 1, package com.example.movielist.data.local.database mendefinisikan package untuk class database Room, bagian dari lapisan data lokal.

Pada baris 3, import androidx.room.Database mengimpor anotasi @Database, yang menandai class sebagai database Room.

Pada baris 4, import androidx.room.RoomDatabase mengimpor class RoomDatabase, superclass dari database Room.

Pada baris 5, import com.example.movielist.data.local.dao.MovieDao mengimpor antarmuka MovieDao.

Pada baris 6, import com.example.movielist.data.local.entities.MovieEntity mengimpor class MovieEntity, entitas yang akan menjadi bagian dari database.

Pada baris 8, @Database(entities = [MovieEntity::class], version = 1, exportSchema = false) menandai class sebagai database Room.

Pada baris 8, entities = [MovieEntity::class] mendaftarkan MovieEntity sebagai entitas yang akan menjadi tabel dalam database ini.

Pada baris 8, version = 1 menentukan versi database. Jika skema database berubah, versi harus di-increment.

Pada baris 8, exportSchema = false menonaktifkan ekspor skema database ke file, cocok untuk pengembangan.

Pada baris 9, abstract class AppDatabase : RoomDatabase() mendefinisikan class abstrak AppDatabase yang mewarisi dari RoomDatabase.

Pada baris 10, abstract fun movieDao(): MovieDao mendefinisikan fungsi abstrak untuk mendapatkan instance MovieDao, yang akan diimplementasikan oleh Room secara otomatis.

Pada baris 12, companion object { ... } adalah objek pendamping class.

Pada baris 13, const val DATABASE_NAME = "tmdb_app_db" mendefinisikan konstanta untuk nama file database.

MovieEntity.kt
Pada baris 1, package com.example.movielist.data.local.entities mendefinisikan package untuk entitas Room, bagian dari lapisan data lokal.

Pada baris 3, import androidx.room.Entity mengimpor anotasi @Entity, yang menandai class data sebagai tabel database Room.

Pada baris 4, import androidx.room.PrimaryKey mengimpor anotasi @PrimaryKey, yang menandai properti sebagai primary key tabel.

Pada baris 5, import com.example.movielist.domain.model.Movie mengimpor class Movie, model domain yang akan dipetakan.

Pada baris 7, @Entity(tableName = "movies") menandai class data ini sebagai entitas Room dan menentukan nama tabel database menjadi "movies".

Pada baris 8, data class MovieEntity(...) mendefinisikan class data MovieEntity, yang akan mewakili satu baris dalam tabel movies.

Pada baris 9, @PrimaryKey val id: Int mendeklarasikan id sebagai primary key untuk tabel, setiap MovieEntity harus memiliki ID unik.

Pada baris 10-15, properti lain seperti title, overview, posterPath, releaseDate, voteAverage, dan popularity adalah kolom-kolom dalam tabel movies.

Pada baris 17, fun toDomainMovie(): Movie mendefinisikan fungsi ekstensi yang mengonversi instance MovieEntity menjadi Movie domain model. Ini digunakan saat membaca data dari database dan menyediakannya ke lapisan domain/presentasi.

Pada baris 27, companion object { ... } adalah objek pendamping class.

Pada baris 28, fun fromDomainMovie(movie: Movie, popularity: Double): MovieEntity mendefinisikan fungsi factory dalam companion object yang mengonversi Movie domain model menjadi MovieEntity yang cocok untuk penyimpanan di Room. popularity disertakan karena merupakan kolom yang disimpan di database.

MovieAppPreferences.kt
Pada baris 1, package com.example.movielist.data.local mendefinisikan nama package dari file Kotlin ini, mengelompokkannya dalam lapisan data lokal.

Pada baris 3, import android.content.Context mengimpor class Context yang menyediakan akses ke sumber daya dan layanan sistem.

Pada baris 4, import android.content.SharedPreferences mengimpor class SharedPreferences, API untuk menyimpan data primitif dalam format key-value pairs.

Pada baris 6, class MovieAppPreferences(context: Context) mendefinisikan class MovieAppPreferences yang bertanggung jawab untuk mengelola SharedPreferences, menerima Context untuk inisialisasi.

Pada baris 8, private val sharedPreferences: SharedPreferences = ... mendeklarasikan properti private untuk instance SharedPreferences.

Pada baris 9, context.getSharedPreferences("tmdb_app_prefs", Context.MODE_PRIVATE) menginisialisasi SharedPreferences dengan nama file "tmdb_app_prefs" dan mode private (hanya bisa diakses oleh aplikasi ini).

Pada baris 11, companion object { ... } adalah objek pendamping yang berisi properti dan fungsi yang terkait dengan class, tetapi tidak memerlukan instance class.

Pada baris 12, private const val KEY_API_KEY = "api_key" mendefinisikan konstanta kunci untuk menyimpan API key.

Pada baris 13, private const val KEY_DARK_MODE = "dark_mode" mendefinisikan konstanta kunci untuk menyimpan status mode gelap.

Pada baris 15, fun saveApiKey(apiKey: String) mendefinisikan fungsi untuk menyimpan API key.

Pada baris 16, sharedPreferences.edit().putString(KEY_API_KEY, apiKey).apply() mengambil editor SharedPreferences, menyimpan string dengan kunci KEY_API_KEY, dan menerapkan perubahan secara asinkron.

Pada baris 19, fun getApiKey(): String? mendefinisikan fungsi untuk mengambil API key yang tersimpan.

Pada baris 20, return sharedPreferences.getString(KEY_API_KEY, null) mengambil string dari SharedPreferences dengan kunci KEY_API_KEY; jika tidak ada, mengembalikan null.

Pada baris 23, fun saveDarkModeState(isDarkMode: Boolean) mendefinisikan fungsi untuk menyimpan status mode gelap (boolean).

Pada baris 24, sharedPreferences.edit().putBoolean(KEY_DARK_MODE, isDarkMode).apply() menyimpan boolean dengan kunci KEY_DARK_MODE.

Pada baris 27, fun getDarkModeState(): Boolean mendefinisikan fungsi untuk mengambil status mode gelap yang tersimpan.

Pada baris 28, return sharedPreferences.getBoolean(KEY_DARK_MODE, false) mengambil boolean dari SharedPreferences dengan kunci KEY_DARK_MODE; jika tidak ada, mengembalikan false (default light mode).

RetrofitClient.kt:
Pada baris 1, package com.example.movielist.data.remote.api mendefinisikan package untuk klien API, bagian dari lapisan data remote.

Pada baris 3, import com.jakewharton.retrofit.retrofit2.kotlinx.serialization.asConverterFactory mengimpor fungsi ekstensi untuk menggunakan KotlinX Serialization dengan Retrofit.

Pada baris 4, import kotlinx.serialization.json.Json mengimpor class Json dari KotlinX Serialization, digunakan untuk mengonfigurasi parser JSON.

Pada baris 5, import okhttp3.MediaType.Companion.toMediaType mengimpor fungsi ekstensi untuk membuat MediaType.

Pada baris 6, import okhttp3.OkHttpClient mengimpor class OkHttpClient, klien HTTP yang akan digunakan Retrofit.

Pada baris 7, import okhttp3.logging.HttpLoggingInterceptor mengimpor interceptor untuk logging permintaan dan respons HTTP.

Pada baris 8, import retrofit2.Retrofit mengimpor class Retrofit, builder utama untuk API service.

Pada baris 9, import java.util.concurrent.TimeUnit mengimpor class TimeUnit untuk mengonfigurasi durasi timeout.

Pada baris 11, object RetrofitClient mendeklarasikan objek singleton RetrofitClient, artinya hanya ada satu instance dari class ini di seluruh aplikasi.

Pada baris 13, private const val BASE_URL = "https://api.themoviedb.org/3/" mendefinisikan URL dasar untuk semua permintaan ke TMDB API.

Pada baris 15, private val json = Json { ... } menginisialisasi instance Json untuk konfigurasi parser JSON.

Pada baris 16, ignoreUnknownKeys = true mengonfigurasi parser untuk mengabaikan kunci JSON yang tidak ada di model data Kotlin Anda, mencegah crash jika ada perubahan di API.

Pada baris 17, prettyPrint = true mengonfigurasi output JSON agar mudah dibaca (berguna untuk debugging).

Pada baris 20, private val okHttpClient: OkHttpClient by lazy { ... } mendeklarasikan instance OkHttpClient secara lazy (dibuat saat pertama kali diakses).

Pada baris 21, val logging = HttpLoggingInterceptor() membuat instance HttpLoggingInterceptor.

Pada baris 22, logging.setLevel(HttpLoggingInterceptor.Level.BODY) mengatur level logging untuk menampilkan header dan body dari permintaan/respons HTTP di Logcat.

Pada baris 24, OkHttpClient.Builder() memulai proses membangun OkHttpClient.

Pada baris 25, .addInterceptor(logging) menambahkan interceptor logging ke klien HTTP.

Pada baris 26, .connectTimeout(30, TimeUnit.SECONDS) mengatur waktu tunggu untuk membuat koneksi.

Pada baris 27, .readTimeout(30, TimeUnit.SECONDS) mengatur waktu tunggu untuk membaca data dari server.

Pada baris 28, .writeTimeout(30, TimeUnit.SECONDS) mengatur waktu tunggu untuk mengirim data ke server.

Pada baris 29, .build() membangun instance OkHttpClient.

Pada baris 32, val tmdbApiService: TmdbApiService by lazy { ... } mendeklarasikan instance TmdbApiService secara lazy.

Pada baris 33, Retrofit.Builder() memulai proses membangun Retrofit.

Pada baris 34, .baseUrl(BASE_URL) menetapkan URL dasar untuk semua permintaan.

Pada baris 35, .client(okHttpClient) menetapkan OkHttpClient kustom yang baru dibuat.

Pada baris 36, .addConverterFactory(json.asConverterFactory("application/json".toMediaType())) menambahkan konverter untuk mengubah JSON menjadi objek Kotlin menggunakan KotlinX Serialization.

Pada baris 37, .build() membangun instance Retrofit.

Pada baris 38, .create(TmdbApiService::class.java) membuat implementasi TmdbApiService dari antarmuka yang didefinisikan.

TmdbApiService.kt:

Pada baris 1, package com.example.movielist.data.remote.api mendefinisikan package untuk antarmuka API, bagian dari lapisan data remote.

Pada baris 3, import com.example.movielist.data.remote.models.MovieListResponse mengimpor class model respons untuk daftar film.

Pada baris 4, import retrofit2.Response mengimpor class Response dari Retrofit untuk membungkus respons HTTP.

Pada baris 5, import retrofit2.http.GET mengimpor anotasi @GET untuk menentukan jenis permintaan HTTP (GET).

Pada baris 6, import retrofit2.http.Query mengimpor anotasi @Query untuk menambahkan parameter kueri ke URL.

Pada baris 8, interface TmdbApiService mendeklarasikan antarmuka TmdbApiService.

Pada baris 10, @GET("movie/popular") menandai fungsi ini untuk melakukan permintaan GET ke endpoint "movie/popular" dari URL dasar TMDB API.

Pada baris 11, suspend fun getPopularMovies(...) mendefinisikan fungsi suspend untuk mendapatkan daftar film populer.

Pada baris 12, @Query("api_key") apiKey: String mendeklarasikan parameter kueri "api_key" yang wajib diisi.

Pada baris 13, @Query("language") language: String = "en-US" mendeklarasikan parameter kueri "language" dengan nilai default "en-US".

Pada baris 14, @Query("page") page: Int = 1 mendeklarasikan parameter kueri "page" dengan nilai default 1 untuk pagination.

Pada baris 15, : Response<MovieListResponse> menentukan bahwa fungsi ini akan mengembalikan objek Response yang membungkus MovieListResponse.

MovieDto.kt:

Pada baris 1, package com.example.movielist.data.remote.models mendefinisikan package untuk model data remote (DTOs), bagian dari lapisan data.

Pada baris 3, import kotlinx.serialization.SerialName mengimpor anotasi @SerialName, digunakan untuk memetakan nama properti Kotlin ke nama kunci JSON yang berbeda.

Pada baris 4, import kotlinx.serialization.Serializable mengimpor anotasi @Serializable, yang menandai class data ini agar dapat di-serialize dan di-deserialize oleh KotlinX Serialization.

Pada baris 6, @Serializable menandai class data ini sebagai class yang bisa diubah menjadi/dari JSON.

Pada baris 7, data class MovieDto(...) mendefinisikan class data MovieDto, yang merupakan Data Transfer Object (DTO) untuk film dari TMDB API. Struktur propertinya mencerminkan struktur JSON yang diterima dari API.

Pada baris 8, val adult: Boolean mendefinisikan properti adult (apakah film ditujukan untuk dewasa).

Pada baris 9, @SerialName("backdrop_path") val backdropPath: String? memetakan kunci JSON "backdrop_path" ke properti backdropPath di Kotlin; tanda ? menunjukkan properti ini bisa null.

Pada baris 11, @SerialName("genre_ids") val genreIds: List<Int> memetakan kunci JSON "genre_ids" ke daftar ID genre.

Pada baris 13, val id: Int mendefinisikan properti id (ID unik film).

Pada baris 14, @SerialName("original_language") val originalLanguage: String memetakan kunci JSON "original_language".

Pada baris 16, @SerialName("original_title") val originalTitle: String memetakan kunci JSON "original_title".

Pada baris 18, val overview: String mendefinisikan properti overview (ringkasan film).

Pada baris 19, val popularity: Double mendefinisikan properti popularity.

Pada baris 20, @SerialName("poster_path") val posterPath: String? memetakan kunci JSON "poster_path" ke properti posterPath (jalur ke gambar poster).

Pada baris 22, @SerialName("release_date") val releaseDate: String memetakan kunci JSON "release_date" ke properti releaseDate (tanggal rilis film).

Pada baris 24, val title: String mendefinisikan properti title (judul film).

Pada baris 25, val video: Boolean mendefinisikan properti video.

Pada baris 26, @SerialName("vote_average") val voteAverage: Double memetakan kunci JSON "vote_average" (rata-rata rating).

Pada baris 28, @SerialName("vote_count") val voteCount: Int memetakan kunci JSON "vote_count" (jumlah voting).

MovieListResponse.kt

Pada baris 1, package com.example.movielist.data.remote.models mendefinisikan package untuk model data remote (DTOs), bagian dari lapisan data.

Pada baris 3, import kotlinx.serialization.SerialName mengimpor anotasi @SerialName.

Pada baris 4, import kotlinx.serialization.Serializable mengimpor anotasi @Serializable.

Pada baris 6, @Serializable menandai class ini agar dapat di-serialize dan di-deserialize oleh KotlinX Serialization.

Pada baris 7, data class MovieListResponse(...) mendefinisikan class data MovieListResponse, yang mewakili struktur respons keseluruhan dari API ketika meminta daftar film (misalnya, endpoint movie/popular akan mengembalikan objek dengan properti seperti page, results, total_pages, total_results).

Pada baris 8, val page: Int mendefinisikan properti untuk nomor halaman saat ini.

Pada baris 9, val results: List<MovieDto> mendefinisikan properti results yang merupakan daftar dari MovieDto, yaitu daftar film yang sebenarnya.

Pada baris 10, @SerialName("total_pages") val totalPages: Int memetakan kunci JSON "total_pages" ke properti totalPages (jumlah total halaman).

Pada baris 12, @SerialName("total_results") val totalResults: Int memetakan kunci JSON "total_results" ke properti totalResults (jumlah total hasil).

MovieDtoExtension.kt:

Pada baris 1, package com.example.movielist.data.remote.models mendefinisikan package dari file ekstensi ini, yang berisi fungsi-fungsi untuk mengubah model data.

Pada baris 3, import com.example.movielist.domain.model.Movie mengimpor class Movie, model domain.

Pada baris 4, import com.example.movielist.data.local.entities.MovieEntity mengimpor class MovieEntity, entitas Room.

Pada baris 6, fun MovieDto.toDomainMovie(): Movie mendefinisikan fungsi ekstensi untuk class MovieDto. Fungsi ini akan mengubah (memetakan) sebuah objek MovieDto (dari API) menjadi Movie domain model. Ini penting untuk menjaga pemisahan lapisan, memastikan bahwa lapisan domain hanya berinteraksi dengan modelnya sendiri.

Pada baris 15, fun MovieDto.toMovieEntity(): MovieEntity mendefinisikan fungsi ekstensi lain untuk class MovieDto. Fungsi ini akan mengubah (memetakan) sebuah objek MovieDto (dari API) menjadi MovieEntity yang dapat disimpan di Room Database.

MovieRepository.kt

Pada baris 1, package com.example.movielist.data.repository mendefinisikan package untuk repository, bagian dari lapisan data.

Pada baris 3-10, import berbagai class dan interface yang dibutuhkan untuk fungsionalitas repository (DAO, API service, model, Flow, Result).

Pada baris 12, interface MovieRepository mendeklarasikan antarmuka MovieRepository. Ini mendefinisikan kontrak tentang bagaimana data film akan disediakan, tanpa mengungkapkan detail implementasinya.

Pada baris 13, fun getPopularMovies(): Flow<Result<List<Movie>>> adalah satu-satunya fungsi yang didefinisikan dalam antarmuka, yang akan mengembalikan Flow yang membungkus Result dari daftar Movie.

Pada baris 16, class MovieRepositoryImpl(...) : MovieRepository mendefinisikan class MovieRepositoryImpl, yang merupakan implementasi konkret dari antarmuka MovieRepository. Ini juga mengambil dependensi TmdbApiService (untuk jaringan) dan MovieDao (untuk database lokal) melalui konstruktornya.

Pada baris 20, override fun getPopularMovies(): Flow<Result<List<Movie>>> = flow { ... } mengimplementasikan fungsi dari antarmuka. Ini adalah inti dari strategi caching dan pengambilan data.

Pada baris 21, emit(Result.Loading) segera memancarkan status Loading ke Flow, memberi tahu UI bahwa proses pengambilan data telah dimulai.

Pada baris 23, val cachedMovies = movieDao.getAllMovies().map { it.toDomainMovie() } mencoba mengambil data film yang sudah ada di cache Room Database. Data ini kemudian dipetakan ke domain model Movie.

Pada baris 24, if (cachedMovies.isNotEmpty()) { emit(Result.Success(cachedMovies)) } Jika ada data di cache, data tersebut segera dipancarkan sebagai Result.Success. Ini memastikan aplikasi dapat menampilkan data dengan cepat (misalnya, dalam mode offline atau saat jaringan lambat) sebelum mencoba dari jaringan.

Pada baris 28, try { ... } catch (...) { ... } adalah blok penanganan kesalahan yang akan mencoba mengambil data dari jaringan dan menangani berbagai jenis error.

Pada baris 29, val response = apiService.getPopularMovies(apiKey = apiKey) melakukan panggilan API ke TMDB untuk mendapatkan daftar film populer terbaru.

Pada baris 30, if (response.isSuccessful) { ... } memeriksa apakah panggilan API berhasil (kode status 2xx).

Pada baris 31, val movieDtos = response.body()?.results ?: emptyList() mengambil daftar DTO film dari body respons; jika null, mengembalikan daftar kosong.

Pada baris 32, val domainMovies = movieDtos.map { it.toDomainMovie() } memetakan DTO yang diterima dari API ke domain model Movie.

Pada baris 34, movieDao.clearAllMovies() menghapus semua data film yang ada di cache Room. Ini adalah bagian dari strategi refresh cache.

Pada baris 35, movieDao.insertAllMovies(movieDtos.map { it.toMovieEntity() }) menyisipkan data film yang baru diterima dari API ke dalam cache Room.

Pada baris 37, emit(Result.Success(domainMovies)) memancarkan data film terbaru yang berhasil diambil dari API ke Flow. Ini akan memperbarui UI dengan data terbaru.

Pada baris 39-47, catch (e: HttpException), catch (e: IOException), dan catch (e: Exception) menangani berbagai jenis kesalahan (HTTP error, masalah koneksi/timeout, atau error umum) dan memancarkan Result.Error yang sesuai.

Movie.kt

Pada baris 1, package com.example.movielist.domain.model mendefinisikan package untuk model domain, bagian dari lapisan domain.

Pada baris 3, import android.os.Parcelable mengimpor antarmuka Parcelable, yang memungkinkan objek ini untuk dikirim antar komponen Android (misalnya antar Activity) secara efisien.

Pada baris 4, import kotlinx.parcelize.Parcelize mengimpor anotasi @Parcelize dari plugin Kotlin Parcelize.

Pada baris 6, @Parcelize adalah anotasi yang secara otomatis menghasilkan implementasi kode Parcelable boilerplate untuk class data ini. Ini menggantikan kebutuhan untuk menulis implementasi Parcelable secara manual.

Pada baris 7, data class Movie(...) : Parcelable mendefinisikan class data Movie. Ini adalah model domain yang bersih dan tidak bergantung pada detail implementasi API (DTO) atau database (Entity). Ia hanya berisi data yang relevan untuk logika bisnis dan presentasi. : Parcelable menandakan bahwa class ini mengimplementasikan antarmuka Parcelable.

Pada baris 8-13, properti seperti id, title, overview, posterPath, releaseDate, dan voteAverage adalah properti dari film yang relevan di lapisan domain.

GetPopularMoviesUseCase.kt

Pada baris 1, package com.example.movielist.domain.usecase mendefinisikan package untuk use case, bagian dari lapisan domain.

Pada baris 3-6, import class yang dibutuhkan (model domain, repository implementasi, Result, Flow).

Pada baris 8, class GetPopularMoviesUseCase(...) mendefinisikan use case GetPopularMoviesUseCase. Use case ini mengkapsulasi logika bisnis spesifik untuk "mendapatkan daftar film populer".

Pada baris 9, private val movieRepository: MovieRepositoryImpl mendeklarasikan dependensi pada implementasi repository (MovieRepositoryImpl). Dalam Clean Architecture yang lebih ketat, ini seharusnya bergantung pada antarmuka MovieRepository dari lapisan domain, namun disesuaikan dengan keputusan Anda untuk menggabungkannya.

Pada baris 11, operator fun invoke(): Flow<Result<List<Movie>>> adalah fungsi operator invoke. Ini memungkinkan instance dari GetPopularMoviesUseCase dipanggil sebagai fungsi (misalnya getPopularMoviesUseCase()) alih-alih getPopularMoviesUseCase.invoke(). Fungsi ini mengembalikan Flow yang membungkus Result dari daftar Movie.

Pada baris 12, return movieRepository.getPopularMovies() memanggil fungsi getPopularMovies() dari repository untuk mendapatkan data, dan mengembalikan Flow hasilnya. Use case ini sendiri tidak memiliki logika kompleks lain selain mendelegasikan tugas ke repository.

DetailActivity.kt

Pada baris 1, package com.example.movielist.presentation.ui.activity mendefinisikan package untuk Activity detail, bagian dari lapisan presentasi.

Pada baris 3-9, import berbagai class dan interface yang dibutuhkan untuk Activity (Bundle, MenuItem, Toast, AppCompatActivity, Glide, dll.).

Pada baris 11, class DetailActivity : AppCompatActivity() mendefinisikan class DetailActivity yang akan menampilkan detail film.

Pada baris 14, private lateinit var binding: ActivityDetailBinding mendeklarasikan variabel binding untuk View Binding.

Pada baris 16, companion object { ... } adalah objek pendamping class.

Pada baris 17, const val EXTRA_MOVIE = "extra_movie" mendefinisikan konstanta kunci yang digunakan untuk meneruskan objek Movie melalui Intent.

Pada baris 20, override fun onCreate(savedInstanceState: Bundle?) adalah method yang dipanggil saat Activity pertama kali dibuat.

Pada baris 21, super.onCreate(savedInstanceState) memanggil implementasi onCreate dari superclass.

Pada baris 22, binding = ActivityDetailBinding.inflate(layoutInflater) menginisialisasi View Binding.

Pada baris 23, setContentView(binding.root) menetapkan layout activity_detail.xml sebagai tampilan Activity.

Pada baris 25, supportActionBar?.setDisplayHomeAsUpEnabled(true) mengaktifkan tombol "Home" (biasanya panah kembali) di ActionBar, memungkinkan navigasi ke Activity sebelumnya.

Pada baris 27-31, val movie = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) { ... } else { ... } mengambil objek Movie yang diteruskan dari MainActivity melalui Intent. Ini menggunakan cara yang berbeda untuk API level Tiramisu (33) ke atas, dan cara yang sudah didepresiasi untuk API level di bawahnya.

Pada baris 33, movie?.let { ... } adalah blok yang akan dieksekusi hanya jika movie tidak null (artinya objek berhasil diterima).

Pada baris 34, supportActionBar?.title = it.title mengatur judul ActionBar menjadi judul film.

Pada baris 36, binding.apply { ... } adalah blok apply untuk mengatur data film ke elemen UI.

Pada baris 37-40, tvDetailTitle.text = it.title, tvDetailReleaseDate.text, tvDetailVoteAverage.text, dan tvDetailOverview.text = it.overview mengisi TextViews dengan data dari objek Movie.

Pada baris 42, val imageUrl = "https://image.tmdb.org/t/p/w500${it.posterPath}" membangun URL lengkap untuk gambar poster film.

Pada baris 43-46, Glide.with(this@DetailActivity).load(imageUrl).centerCrop().into(ivDetailPoster) menggunakan Glide untuk memuat gambar poster ke ImageView.

Pada baris 48-51, ?: run { ... } adalah operator Elvis yang akan dieksekusi jika movie adalah null; ini menampilkan Toast dan menutup Activity.

Pada baris 54, override fun onOptionsItemSelected(item: MenuItem): Boolean adalah method yang dipanggil ketika item di ActionBar diklik.

Pada baris 55, if (item.itemId == android.R.id.home) memeriksa apakah item yang diklik adalah tombol "Home" (tombol back).

Pada baris 56, onBackPressedDispatcher.onBackPressed() mensimulasikan penekanan tombol back perangkat, membawa pengguna kembali ke Activity sebelumnya.

MovieAdapter.kt

Pada baris 1, package com.example.movielist.presentation.ui.adapter mendefinisikan package untuk adapter RecyclerView, bagian dari lapisan presentasi.

Pada baris 3-8, import berbagai class yang dibutuhkan (LayoutInflater, ViewGroup, DiffUtil, ListAdapter, RecyclerView, Glide, Binding, model domain).

Pada baris 10, class MovieAdapter : ListAdapter<Movie, MovieAdapter.MovieViewHolder>(MovieDiffCallback()) mendefinisikan MovieAdapter. Ini adalah ListAdapter, yang merupakan jenis adapter RecyclerView yang efisien dalam memperbarui daftar item. Ia menerima Movie sebagai tipe data dan MovieViewHolder sebagai ViewHolder. MovieDiffCallback() digunakan untuk membandingkan item secara efisien.

Pada baris 12, var onItemClick: ((Movie) -> Unit)? = null mendeklarasikan sebuah properti lambda nullable bernama onItemClick. Ini akan berfungsi sebagai callback

yang dapat diatur dari MainActivity untuk menangani klik pada tombol "Detail" di setiap item.

Pada baris 14, override fun onCreateViewHolder(...) membuat dan mengembalikan instance MovieViewHolder. Ini meng-inflate layout item_movie.xml menggunakan View Binding.

Pada baris 19, override fun onBindViewHolder(...) mengikat data Movie ke ViewHolder pada posisi tertentu di daftar.

Pada baris 24, inner class MovieViewHolder(...) mendefinisikan inner class MovieViewHolder yang memegang referensi ke tampilan setiap item di RecyclerView.

Pada baris 27, init { binding.btnDetail.setOnClickListener { ... } } adalah blok inisialisasi untuk ViewHolder. Di sinilah OnClickListener untuk tombol "Detail" diatur.

Pada baris 28, onItemClick?.invoke(getItem(adapterPosition)) memanggil lambda onItemClick yang telah diatur dari MainActivity, meneruskan objek Movie yang sesuai dengan posisi item yang diklik. Tanda ? memastikan invoke hanya dipanggil jika onItemClick tidak null.

Pada baris 32, fun bind(movie: Movie) mendefinisikan fungsi bind yang bertanggung jawab untuk mengisi tampilan item dengan data dari objek Movie.

Pada baris 33, binding.apply { ... } menggunakan fungsi scope apply untuk bekerja dengan properti binding.

Pada baris 34-37, tvMovieTitle.text, tvReleaseDate.text, tvVoteAverage.text, tvOverview.text mengisi TextViews dengan data dari movie.

Pada baris 39, val imageUrl = "https://image.tmdb.org/t/p/w500${movie.posterPath}" membangun URL gambar poster.

Pada baris 40-43, Glide.with(itemView.context).load(imageUrl).centerCrop().into(ivPoster) menggunakan Glide untuk memuat gambar poster ke ImageView.

Pada baris 46, class MovieDiffCallback : DiffUtil.ItemCallback<Movie>() mendefinisikan DiffUtil.ItemCallback kustom. Ini digunakan oleh ListAdapter untuk

menghitung perbedaan antara daftar lama dan baru, sehingga RecyclerView dapat diperbarui secara efisien.

Pada baris 47, override fun areItemsTheSame(...) memeriksa apakah dua item mewakili objek yang sama (biasanya dengan membandingkan ID unik mereka).

Pada baris 50, override fun areContentsTheSame(...) memeriksa apakah konten dari dua item yang sama persis juga sama (digunakan untuk mendeteksi perubahan dalam data item).

MovieViewModel.kt

Pada baris 1, package com.example.movielist.presentation.viewmodel mendefinisikan package untuk ViewModel, bagian dari lapisan presentasi.

Pada baris 3-9, import berbagai class yang dibutuhkan (LiveData, ViewModel, ViewModelScope, model domain, use case, Result, Flow, Coroutines).

Pada baris 11, class MovieViewModel(...) : ViewModel() mendefinisikan MovieViewModel, yang merupakan turunan dari androidx.lifecycle.ViewModel. ViewModel bertanggung jawab untuk menyiapkan dan mengelola data yang terkait dengan UI agar data tetap ada saat konfigurasi perangkat berubah (misalnya, rotasi layar).

Pada baris 12, private val getPopularMoviesUseCase: GetPopularMoviesUseCase mendeklarasikan dependensi pada GetPopularMoviesUseCase. ViewModel tidak berinteraksi langsung dengan repository atau API, tetapi mendelegasikan logika bisnis ke use case.

Pada baris 15, private val _popularMovies = MutableLiveData<Result<List<Movie>>>() mendeklarasikan MutableLiveData private. Ini adalah LiveData yang dapat diubah nilainya. Ini akan menampung hasil pengambilan data film (berupa Result yang membungkus daftar Movie).

Pada baris 16, val popularMovies: LiveData<Result<List<Movie>>> = _popularMovies mendeklarasikan LiveData publik yang tidak dapat diubah (immutable). UI akan mengamati LiveData ini untuk mendapatkan pembaruan data.

Pada baris 18, init { fetchPopularMovies() } adalah blok inisialisasi yang akan dipanggil saat instance MovieViewModel pertama kali dibuat. Ini memicu pengambilan data film.

Pada baris 21, fun fetchPopularMovies() mendefinisikan fungsi untuk memicu pengambilan data film.

Pada baris 22, viewModelScope.launch { ... } meluncurkan coroutine dalam cakupan viewModelScope. viewModelScope memastikan bahwa coroutine ini akan dibatalkan secara otomatis ketika ViewModel dihancurkan, mencegah kebocoran memori.

Pada baris 23, getPopularMoviesUseCase().collect { result -> ... } memanggil invoke() operator dari use case dan mengumpulkan nilai-nilai yang dipancarkan oleh Flow yang dikembalikan oleh use case. Setiap kali use case memancarkan Result baru (Loading, Success, atau Error), blok collect akan menerimanya.

Pada baris 24, _popularMovies.value = result memperbarui nilai MutableLiveData. Perubahan ini akan secara otomatis memberitahu Observer di UI (MainActivity) untuk memperbarui tampilannya.

ViewModelFactory.kt

Pada baris 1, package com.example.movielist.presentation.viewmodel mendefinisikan package untuk ViewModel Factory, bagian dari lapisan presentasi.

Pada baris 3, import androidx.lifecycle.ViewModel mengimpor class ViewModel.

Pada baris 4, import androidx.lifecycle.ViewModelProvider mengimpor class ViewModelProvider, yang digunakan untuk membuat instance ViewModel.

Pada baris 5, import com.example.movielist.domain.usecase.GetPopularMoviesUseCase mengimpor use case.

Pada baris 7, class ViewModelFactory(...) : ViewModelProvider.Factory mendefinisikan ViewModelFactory kustom yang mengimplementasikan ViewModelProvider.Factory. Factory ini bertanggung jawab untuk membuat instance ViewModel dengan dependensi yang diperlukan, karena ViewModel tidak dapat memiliki konstruktor dengan parameter secara langsung oleh sistem Android.

Pada baris 8, private val getPopularMoviesUseCase: GetPopularMoviesUseCase adalah dependensi yang dibutuhkan oleh MovieViewModel. Factory ini menerimanya melalui konstruktor.

Pada baris 11, override fun <T : ViewModel> create(modelClass: Class<T>): T adalah method yang harus diimplementasikan dari ViewModelProvider.Factory. Method ini bertanggung jawab untuk membuat instance ViewModel yang diminta.

Pada baris 12, if (modelClass.isAssignableFrom(MovieViewModel::class.java)) memeriksa apakah modelClass yang diminta adalah MovieViewModel.

Pada baris 14, return MovieViewModel(getPopularMoviesUseCase) as T jika ViewModel yang diminta adalah MovieViewModel, maka instance baru MovieViewModel dibuat dengan dependensi getPopularMoviesUseCase yang disuntikkan. as T adalah unsafe cast yang di-suppress.

Pada baris 17, throw IllegalArgumentException("Unknown ViewModel class") melempar pengecualian jika modelClass yang diminta tidak dikenali oleh factory ini.

Result.kt

Pada baris 1, package com.example.movielist.utils mendefinisikan package untuk utilitas umum.

Pada baris 3, sealed class Result<out T> mendefinisikan sealed class bernama Result. Sealed class adalah class yang nilai-nilainya terbatas pada satu set subclass yang didefinisikan dalam class itu sendiri. Ini sangat berguna untuk merepresentasikan state yang berbeda (seperti Loading, Success, Error) dengan cara yang aman dan type-safe. out T menunjukkan bahwa T adalah tipe kovarian, artinya Result<Subtype> adalah subclass dari Result<Supertype>.

Pada baris 4, object Loading : Result<Nothing>() adalah objek singleton yang merepresentasikan status data sedang dimuat. Nothing menunjukkan bahwa tidak ada data yang terkait dengan state ini.

Pada baris 5, data class Success<out T>(val data: T) : Result<T>() adalah class data yang merepresentasikan status data berhasil dimuat. Ia membungkus data aktual (val data: T).

Pada baris 6, data class Error(val exception: Exception) : Result<Nothing>() adalah class data yang merepresentasikan status terjadi kesalahan. Ia membungkus objek Exception yang menjelaskan kesalahan tersebut.

Activity_main.xml

Pada baris 1, <?xml version="1.0" encoding="utf-8"?> adalah deklarasi standar XML.

Pada baris 2, <androidx.constraintlayout.widget.ConstraintLayout ...> adalah tag root untuk layout ini. ConstraintLayout adalah layout yang fleksibel yang memungkinkan Anda memposisikan dan mengukur tampilan secara relatif satu sama lain atau ke parent layout.

Pada baris 3-5, xmlns:android, xmlns:app, dan xmlns:tools mendeklarasikan namespace yang berbeda untuk atribut layout.

Pada baris 6, android:layout_width="match_parent" dan android:layout_height="match_parent" membuat layout mengisi seluruh lebar dan tinggi layar.

Pada baris 7, tools:context=".presentation.ui.activity.MainActivity" adalah atribut tools untuk Android Studio, membantu dalam pratinjau layout.

Pada baris 9, <TextView ...> mendeklarasikan TextView untuk judul "Popular Movies".

Pada baris 10, android:id="@+id/tv_title" memberikan ID unik untuk TextView ini.

Pada baris 11-15, mengatur teks, ukuran, gaya, margin, dan constraint posisinya di pojok kiri atas parent.

Pada baris 18, <ProgressBar ...> mendeklarasikan ProgressBar untuk indikator loading.

Pada baris 19, android:id="@+id/progress_bar" memberikan ID unik.

Pada baris 20, android:visibility="gone" menyembunyikan ProgressBar secara default.

Pada baris 21-24, mengatur constraint posisi ProgressBar di tengah layar.

Pada baris 26, <TextView ...> mendeklarasikan TextView untuk pesan error.

Pada baris 27, android:id="@+id/tv_error" memberikan ID unik.

Pada baris 28-30, mengatur teks, warna, visibilitas (default gone), dan perataan teks.

Pada baris 31, app:layout_constraintVertical_chainStyle="packed" dan baris 32-35, mengatur constraint yang membentuk rantai vertikal dengan tombol "Retry", memastikan keduanya terpusat sebagai grup.

Pada baris 37, <Button ...> mendeklarasikan tombol "Retry".

Pada baris 38, android:id="@+id/btn_retry" memberikan ID unik.

Pada baris 39, android:visibility="gone" menyembunyikan tombol secara default.

Pada baris 40-44, mengatur teks, margin, dan constraint posisi di bawah tv_error dan terpusat horizontal.

Pada baris 46, <androidx.recyclerview.widget.RecyclerView ...> mendeklarasikan RecyclerView untuk menampilkan daftar film.

Pada baris 47, android:id="@+id/rv_movies" memberikan ID unik.

Pada baris 48-52, mengatur lebar, tinggi, margin, dan constraint posisi RecyclerView agar mengisi sisa ruang di bawah judul.

Pada baris 53, tools:listitem="@layout/item_movie" adalah atribut tools untuk Android Studio, menampilkan pratinjau item layout di RecyclerView.

item_movie.xml

Pada baris 1, <?xml version="1.0" encoding="utf-8"?> adalah deklarasi standar XML.

Pada baris 2, <androidx.cardview.widget.CardView ...> adalah tag root untuk layout item ini. CardView digunakan untuk memberikan tampilan item dengan sudut membulat dan elevasi (bayangan), yang umum di Material Design.

Pada baris 3-5, xmlns:android, xmlns:app, dan xmlns:tools mendeklarasikan namespace.

Pada baris 6, android:layout_width="match_parent" membuat lebar CardView mengisi lebar parent.

Pada baris 7, android:layout_height="wrap_content" membuat tinggi CardView pas dengan kontennya.

Pada baris 8, android:layout_margin="8dp" menambahkan margin 8dp di semua sisi CardView, memberikan ruang antar item.

Pada baris 9, app:cardCornerRadius="8dp" mengatur radius sudut CardView menjadi 8dp.

Pada baris 10, app:cardElevation="4dp" memberikan bayangan (elevasi) pada CardView.

Pada baris 12, <androidx.constraintlayout.widget.ConstraintLayout ...> adalah layout di dalam CardView, digunakan untuk mengatur posisi elemen-elemen detail film.

Pada baris 13, android:padding="16dp" menambahkan padding 16dp di dalam ConstraintLayout.

Pada baris 15, <ImageView ...> mendeklarasikan ImageView untuk menampilkan poster film.

Pada baris 16, android:id="@+id/iv_poster" memberikan ID unik.

Pada baris 17-20, mengatur lebar, tinggi, skala gambar, dan constraint posisinya di pojok kiri atas layout.

Pada baris 22, <TextView ...> mendeklarasikan TextView untuk judul film.

Pada baris 23, android:id="@+id/tv_movie_title" memberikan ID unik.

Pada baris 24-29, mengatur lebar, tinggi, margin, gaya teks, ukuran, dan constraint posisinya di kanan poster.

Pada baris 31, <TextView ...> mendeklarasikan TextView untuk tanggal rilis.

Pada baris 32, android:id="@+id/tv_release_date" memberikan ID unik.

Pada baris 33-38, mengatur lebar, tinggi, margin, ukuran teks, dan constraint posisinya di bawah judul.

Pada baris 40, <TextView ...> mendeklarasikan TextView untuk rata-rata voting/rating.

Pada baris 41, android:id="@+id/tv_vote_average" memberikan ID unik.

Pada baris 42-47, mengatur lebar, tinggi, margin, ukuran teks, dan constraint posisinya di bawah tanggal rilis.

Pada baris 49, <TextView ...> mendeklarasikan TextView untuk ringkasan (overview) film.

Pada baris 50, android:id="@+id/tv_overview" memberikan ID unik.

Pada baris 51-56, mengatur lebar, tinggi, margin, batas baris (maxLines), elipsis jika teks terlalu panjang, dan constraint posisinya di bawah poster.

Pada baris 58, <Button ...> mendeklarasikan tombol "Detail".

Pada baris 59, android:id="@+id/btn_detail" memberikan ID unik.

Pada baris 60-63, mengatur lebar, tinggi, margin, teks tombol, dan constraint posisinya di pojok kanan

activity_detail.xml

Pada baris 1, <?xml version="1.0" encoding="utf-8"?> adalah deklarasi standar XML.

Pada baris 2, <ScrollView ...> adalah tag root layout. ScrollView memungkinkan konten di dalamnya untuk digulir jika ukurannya melebihi tinggi layar, yang penting untuk detail film/serial yang mungkin panjang.

Pada baris 3-5, xmlns:android, xmlns:app, dan xmlns:tools mendeklarasikan namespace.

Pada baris 6, android:layout_width="match_parent" dan android:layout_height="match_parent" membuat ScrollView mengisi seluruh layar.

Pada baris 7, tools:context=".presentation.ui.activity.DetailActivity" adalah atribut tools untuk Android Studio.

Pada baris 9, <androidx.constraintlayout.widget.ConstraintLayout ...> adalah layout di dalam ScrollView, digunakan untuk mengatur posisi elemen-elemen detail.

Pada baris 10, android:padding="16dp" menambahkan padding di dalam ConstraintLayout.

Pada baris 12, <ImageView ...> mendeklarasikan ImageView untuk menampilkan poster detail.

Pada baris 13, android:id="@+id/iv_detail_poster" memberikan ID unik.

Pada baris 14-18, mengatur lebar (0dp mengisi parent), tinggi, skala, dan constraint posisinya di bagian atas layout.

Pada baris 20, <TextView ...> mendeklarasikan TextView untuk judul film.

Pada baris 21, android:id="@+id/tv_detail_title" memberikan ID unik.

Pada baris 22-27, mengatur lebar, tinggi, margin, ukuran teks, gaya teks, dan constraint posisinya di bawah poster.

Pada baris 29, <TextView ...> mendeklarasikan TextView untuk tanggal rilis.

Pada baris 30, android:id="@+id/tv_detail_release_date" memberikan ID unik.

Pada baris 31-36, mengatur lebar, tinggi, margin, ukuran teks, dan constraint posisinya di bawah judul.

Pada baris 38, <TextView ...> mendeklarasikan TextView untuk rata-rata voting/rating.

Pada baris 39, android:id="@+id/tv_detail_vote_average" memberikan ID unik.

Pada baris 40-45, mengatur lebar, tinggi, margin, ukuran teks, dan constraint posisinya di bawah tanggal rilis.

Pada baris 47, <TextView ...> mendeklarasikan TextView sebagai label "Overview:".

Pada baris 48, android:id="@+id/tv_detail_overview_label" memberikan ID unik.

Pada baris 49-54, mengatur lebar, tinggi, margin, teks, ukuran teks, gaya teks, dan constraint posisinya di bawah rating.

Pada baris 56, <TextView ...> mendeklarasikan TextView untuk teks overview sebenarnya.

Pada baris 57, android:id="@+id/tv_detail_overview" memberikan ID unik.

Pada baris 58-63, mengatur lebar, tinggi, margin, ukuran teks, dan constraint posisinya di bawah label overview.

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Easydaf/Praktikum_Mobile