LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Rifky Putra Mahardika **NIM. 2310817210023**

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 : Rifky Putra Mahardika
NIM : 2310817210023

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar	Muti`a Maulida S.Kom M.T.I
NIM. 2210817210026	NIP. 198810272019032013

DAFTAR ISI

DAFTAR GAMBAR

MODUL 1 : Android Basic with Kotlin

MODUL 2 : Android Layout

MODUL 3 : Build a Scrollable List

MODUL 4 : ViewModel and Debugging

MODUL 5 : Connect to the Internet

DAFTAR TABEL

MODUL 1 : Android Basic with Kotlin

MODUL 2 : Android Layout

MODUL 3 : Build a Scrollable List

MODUL 4 : ViewModel and Debugging

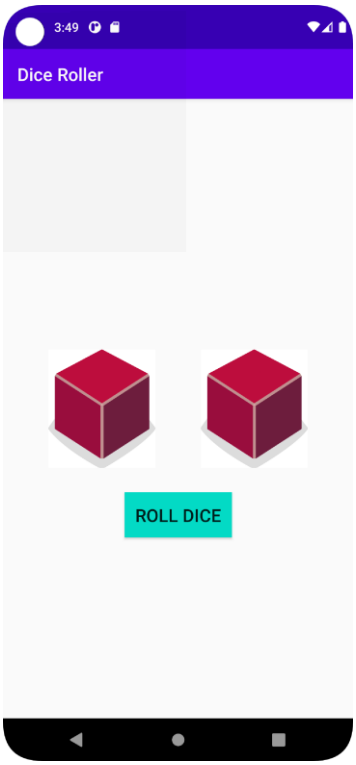
MODUL 5 : Connect to the Internet

MODUL 1 : Android Basic with Kotlin

SOAL 1

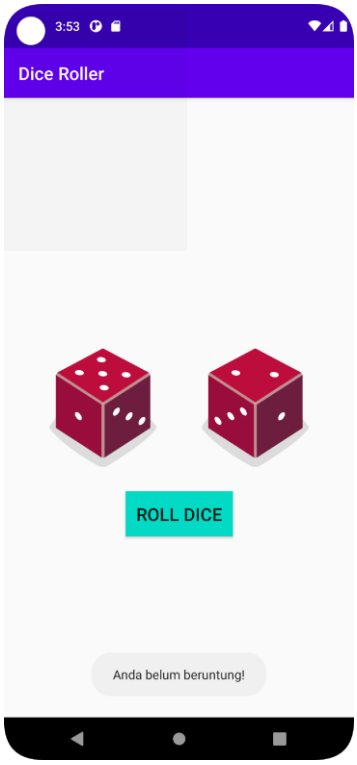
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



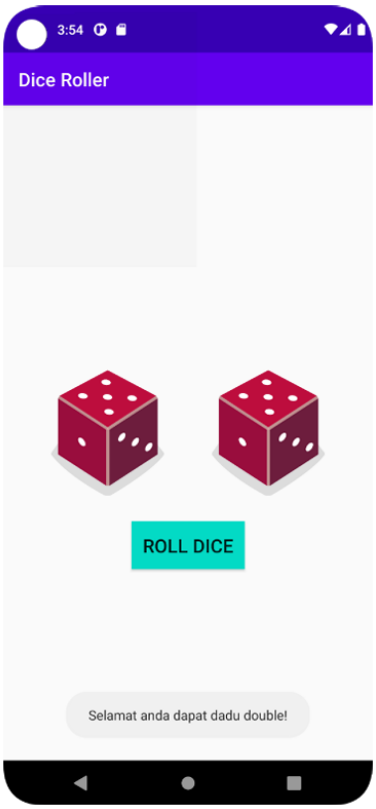
Gambar 1. Contoh Tampilan Awal Aplikasi

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



Gambar 2. Contoh 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 dapat dilihat pada Gambar 3.



Gambar 3. Contoh Tampilan Roll Dadu Double

4. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 2 dalam bentuk project.** Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.

5. Untuk gambar dadu dapat didownload pada link berikut: https://drive.google.com/file/d/14V3qXGdFnuoYN4AGd_9SgFh8kw8X9ySm/view

A. Source Code

1. MainActivity.kt

```
1   package com.example.prakmodul1
2
3   import android.os.Bundle
4   import android.widget.Toast
5   import androidx.activity.ComponentActivity
6   import androidx.activity.compose.setContent
7   import androidx.compose.foundation.Image
8   import androidx.compose.foundation.layout.*
9   import
    androidx.compose.foundation.shape.RoundedCornerShape
10  import androidx.compose.material3.*
11  import androidx.compose.runtime.*
12  import
    androidx.compose.runtime.saveable.rememberSaveable
13  import androidx.compose.ui.Alignment
14  import androidx.compose.ui.Modifier
15  import androidx.compose.ui.graphics.Color
16  import androidx.compose.ui.platform.LocalContext
17  import androidx.compose.ui.res.painterResource
18  import androidx.compose.ui.unit.dp
19  import com.example.prakmodul1.ui.theme.Prakmodul1Theme
20
21  class MainActivity : ComponentActivity() {
22      companion object {
23          private const val pesandouble = "Selamat anda
    dapat dadu double!"
24          private const val kurangberuntung = "Anda belum
    beruntung!"
25      }
26
27      override fun onCreate(savedInstanceState: Bundle?) {
28          super.onCreate(savedInstanceState)
29          setContent {
30              Prakmodul1Theme {
31                  DiceRollerApp()
32              }
33          }
34      }
35
36      @OptIn(ExperimentalMaterial3Api::class)
37      @Composable
38      fun DiceRollerApp() {
39          var dice1 by rememberSaveable {
    mutableStateOf(0) }
40          var dice2 by rememberSaveable {
    mutableStateOf(0) }
41
42          val context = LocalContext.current
43
```

```
44          Scaffold(
45              topBar = {
46                  TopAppBar(
47                      title = {
48                          Text(
49                              text = "Dice Roller",
50                              style =
    MaterialTheme.typography.headlineMedium
51                          )
52                      },
53                      colors =
    TopAppBarDefaults.topAppBarColors(
54                          containerColor =
    Color(0xFF6200EE),
55                          titleContentColor = Color.White
56                      )
57                  )
58              }
59          ) { innerPadding ->
60              Column(
61                  modifier = Modifier
62                      .fillMaxSize()
63                      .padding(innerPadding)
64                      .padding(16.dp),
65                  horizontalAlignment =
    Alignment.CenterHorizontally,
66                  verticalArrangement = Arrangement.Center
67              ) {
68                  Row(
69                      horizontalArrangement =
    Arrangement.spacedBy(32.dp),
70                      verticalAlignment =
    Alignment.CenterVertically
71                  ) {
72                      DiceImage(diceValue = dice1)
73                      DiceImage(diceValue = dice2)
74                  }
75
76                  Spacer(modifier =
    Modifier.height(48.dp))
77
78                  Button(
79                      onClick = {
80                          dice1 = (1..6).random()
81                          dice2 = (1..6).random()
82
83                          val message = if (dice1 ==
    dice2) pesandouble else kurangberuntung
84                          Toast.makeText(context, message,
```

```
    Toast.LENGTH_SHORT).apply {
85  view?.setBackgroundResource(android.R.color.transparent)
86                          }.show()
87                      },
88                      shape = RoundedCornerShape(0.dp),
89                      colors =
    ButtonDefaults.buttonColors(
90                          containerColor =
    Color(0xFF00BCD4),
91                          contentColor = Color.White
92                      ),
93                      modifier = Modifier
94                          .width(200.dp)
95                          .height(50.dp)
96                  ) {
97                      Text(
98                          text = "ROLL DICE",
99                          style =
    MaterialTheme.typography.labelLarge
100                     )
101                 }
102             }
103         }
104     }
105
106     @Composable
107     fun DiceImage(diceValue: Int) {
108         val imageRes = when (diceValue) {
109             0 -> R.drawable.dice_0
110             1 -> R.drawable.dice_1
111             2 -> R.drawable.dice_2
112             3 -> R.drawable.dice_3
113             4 -> R.drawable.dice_4
114             5 -> R.drawable.dice_5
115             else -> R.drawable.dice_6
116         }
117
118         Image(
119             painter = painterResource(id = imageRes),
120             contentDescription = "Dice showing
    $diceValue",
121             modifier = Modifier.size(100.dp)
122         )
123     }
124 }
```

Tabel 1. Source Code Jawaban Soal 1 MainActivity.kt

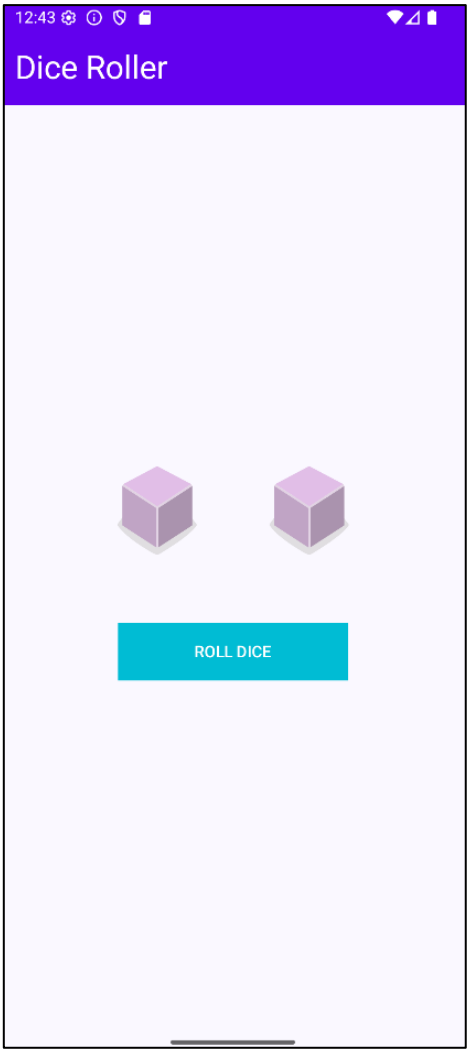
2. activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout
3   xmlns:android="http://schemas.android.com/apk/res/androi
    d"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:orientation="vertical"
8       android:gravity="center"
9       android:padding="16dp"
10      tools:context=".MainActivity">
11
12      <LinearLayout
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:orientation="horizontal"
16          android:layout_marginBottom="32dp">
17
18          <ImageView
19              android:id="@+id/dice1"
20              android:layout_width="100dp"
21              android:layout_height="100dp"
22              android:src="@drawable/dice_0"
23              android:layout_marginEnd="16dp"/>
24
25          <ImageView
26              android:id="@+id/dice2"
27              android:layout_width="100dp"
28              android:layout_height="100dp"
29              android:src="@drawable/dice_0"/>
30      </LinearLayout>
31
32      <TextView
33          android:id="@+id/resultText"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content"
36          android:layout_marginBottom="16dp"
37          android:textSize="18sp"
38          android:textStyle="bold"/>
39
40      <Button
41          android:id="@+id/Button"
42          android:layout_width="wrap_content"
43          android:layout_height="wrap_content"
44          android:text="Roll Dice"
45          android:textAllCaps="false"
46          android:paddingHorizontal="32dp"
```

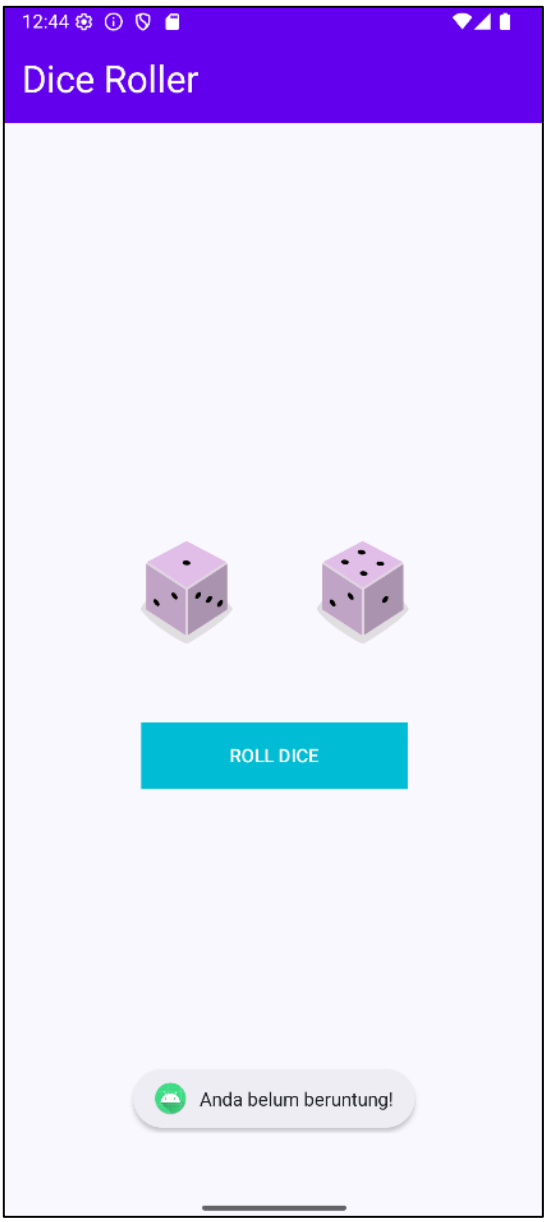
```
47          android:paddingVertical="8dp"/>
48
49  </LinearLayout>
```

Tabel 2. Source Code Jawaban Soal 1 activity_main.xml

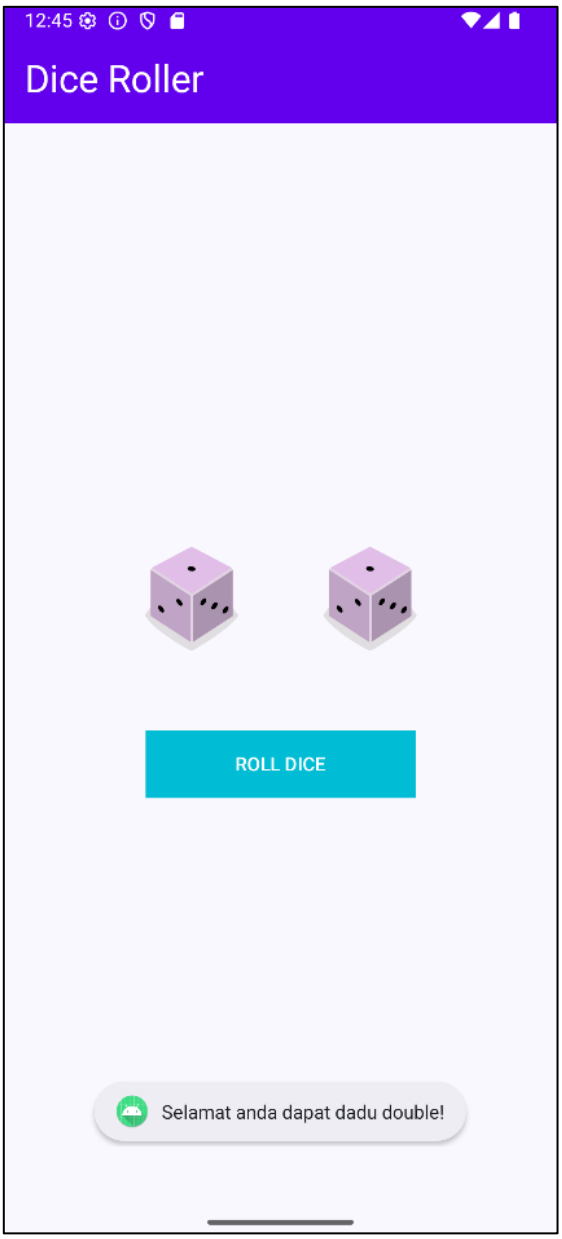
B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 Tampilan Awal



Gambar 5. Screenshot Hasil Jawaban Soal 1 Tidak Double



Gambar 6. Screenshot Hasil Jawaban Soal 1 Double

Gambar 7. Screenshot Hasil Jawaban Soal 1 Mode Landscape

C. Pembahasan

1. MainActivity.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `com.example.prakmodul1`.
- Pada baris [3] hingga [19], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [21], `class MainActivity : ComponentActivity()` ini digunakan sebagai titik awal aplikasi yang akan mengatur tampilan aplikasi.
- Pada baris [22], `companion object` digunakan untuk menyimpan anggota (variabel/fungsi) yang bersifat statis.
- Pada baris [23] dan [24], `private const val pesandouble = "Selamat anda dapat dadu double!"` nantinya digunakan untuk menyimpan teks yang akan ditampilkan ketika hasil kedua dadu nilainya sama atau double. private const val kurangberuntung = "Anda belum beruntung!" ini nantinya untuk menyimpan teks yang akan ditampilkan ketika kedua dadu tidak sama.

- Pada baris [27], `override fun onCreate(savedInstanceState: Bundle?)` berfungsi untuk menimpa fungsi onCreate dari `ComponentActivity`.
- Pada baris [28], `super.onCreate(savedInstanceState)` berfungsi untuk memastikan fungsi bawaan Android tetap dijalankan dengan benar, seperti pelacakan lifecycle dan manajemen sistem.
- Pada baris [29] hingga [31], `setContent{…}` digunakan untuk menetapkan tampilan UI dari aplikasi, dan diterapkan tampilan dari `Prakmodul1Theme`, dan `DiceRollerApp()` yang berisi UI pada dadu.
- Pada baris [36], `@OptIn(ExperimentalMaterial3Api::class)` merupakan penggunaan fitur yang masih bersifat eksperimental di Jetpack Compose Material 3.
- Pada baris [38] hingga [42], `fun DiceRollerApp()` akan akan menampilkan antarmuka aplikasi dadu ini, lalu `var dice1 by rememberSaveable { mutableStateOf(0)` untuk memastikan nilainya tetap bertahan walau terjadi perubahan konfigurasi (seperti rotasi layar). `val context = LocalContext.current` digunakan untuk mengambil konteks saat ini dari lingkungan Compose.
- Pada baris [44], `Scaffold` adalah komponen layout utama di Jetpack Compose yang menyediakan struktur standar UI aplikasi.
- Pada baris [45] hingga [50], `topBar` Menentukan konten yang akan ditampilkan di bagian atas layar dan ditetapkan judul Dice Roller pada bagian `topBar` tersebut.
- Pada baris [53] hingga [55], `colors = TopAppBarDefaults.topAppBarColors(...)` ini digunakan untuk mengatur warna dari app bar tadi.

- Pada baris [59], `innerPadding` adalah penutup dari `Scaffold` untuk menghindari tumpang tindih dengan sistem UI.
- Pada baris [60] hingga [66], `Column` adalah layout vertikal di Compose dan Semua elemen di dalamnya akan ditumpuk dari atas ke bawah. `Modifier` digunakan untuk mengatur ukuran dan tampilan komponen.
- Pada baris [68] hingga [70], `Row` digunakan untuk mengatur layout horizontal pada Compose dan diatur ukuran dan tampilan komponennya pada baris ini.
- Pada baris [72] dan [73], `DiceImage` digunakan untuk menampilkan gambar dan angka dari kedua dadu tersebut.
- Pada baris [78], `Button` merupakan komponen tombol dari Jetpack Compose dan dapat diedit sendiri.
- Pada baris [79] hingga [81], `onClick` merupakan proses yang dijalankan pada saat tombol diklik dan dadu akan dirandom angkanya.
- Pada baris [83], `val message = if (dice1 == dice2) pesandouble else kurangberuntung` digunakan untuk menampilkan pesan dadu double maupun tidak double.
- Pada baris [84], Toast.makeText(...) digunakan untuk menampilkan pesan singkat di layar berdasarkan kondisi.
- Pada baris [88] hingga [91], shape = RoundedCornerShape(0.dp) digunakan untuk mengatur tombol menjadi bentuk persegi, dan selanjutkan diatur warna pada tombol.
- Pada baris [93], `modifier = Modifier …` digunakan untuk mengatur ukuran tombol.
- Pada baris [97] hingga [99], Text digunakan untuk menampilkan **teks** di dalam UI aplikasi.

- Pada baris [106], `@Composable` merupakan anotasi dari Compose, yang dimana ini sebagai penanda bahwa `DiceImage` adalah fungsi yang bisa digunakan dalam UI deklaratif Jetpack Compose.
- Pada baris [107], `fun DiceImage(diceValue: Int)` ini nantinya akan menerima satu nilai dadu dari 0 hingga 6 dalam bentuk integer dan menentukan gambar mana yang akan tampil nanti.
- Pada baris [109] hingga [115], akan ditampilkan gambar dadu sesuai dengan nilai dadu yang sudah diacak sebelumnya.
- Pada baris [118] hingga [121], `Image` merupakan komponen yang nantinya akan menampilkan gambar pada output. `painter = painterResource(id = imageRes)` akan mengambil gambar dari resource drawable berdasarkan `imageRes`. `contentDescription = "Dice showing $diceValue"` sebagai deskripsi gambar bagi pengguna.

2. activity_main.xml

- Pada baris [1], `<?xml version="1.0" encoding="utf-8"?>` ini digunakan sebagai deklarasi XML dengan menggunakan XML versi 1.0 dengan encoding UTF-8.
- Pada baris [2] hingga [9], `LinearLayout` ini digunakan sebagai container utama dengan orientasi vertikal (komponen ditata dari atas ke bawah). `xmlns:android` merupakan namespace standar untuk atribut Android. Untuk selanjutnya diatur lebar layar, tinggi layar, layout akan disejajarkan di Tengah, dan lain sebagainya.
- Pada baris [11] hingga [15], `LinearLayout` ini digunakan sebagai container untuk dua dadu.
- Pada baris [17] hingga [22], `ImageView` ini nantinya digunakan untuk menampilkan gambar dadu, yang dimana pada bagian ini untuk mengatur

tampilan pada dadu pertama. Pada bagian ini diatur sumber gambar dadu, lebar, dan juga tinggi dadu.

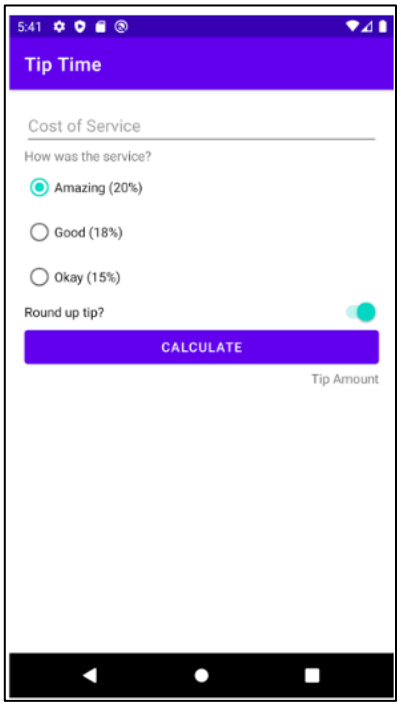
- Pada baris [24] hingga [28], `ImageView` ini nantinya digunakan untuk menampilkan gambar dadu, yang dimana pada bagian ini untuk mengatur tampilan pada dadu kedua. Pada bagian ini diatur sumber gambar dadu, lebar, dan juga tinggi dadu.
- Pada baris [31] hingga [37], `TextView` ini nantinya akan digunakan untuk menampilkan hasil dari roll dadu dan diatur lebar, tinggi, ketebalan, dan ukuran teks dari hasil roll dadu ini.
- Pada baris [39] hingga [46], `Button` merupakan tombol yang nantinya akan digunakan untuk melakukan roll dadu pada saat tombol ditekan dan diatur pula teks pada tombol ini yaitu “Roll Dice”.
- Pada baris [48], `</LinearLayout>` digunakan untuk menutup LinearLayout utama dari file XML ini.

MODUL 2 : Android Layout

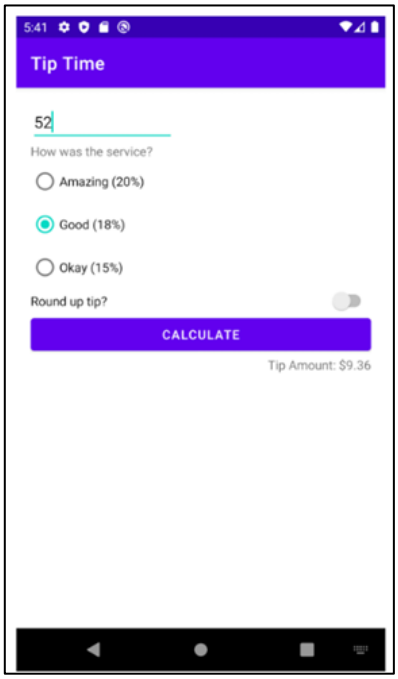
SOAL 1

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

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



Gambar 8. Tampilan Awal Aplikasi



Gambar 9. Tampilan Aplikasi Setelah Dijalankan

A. Source Code

1. MainActivity.kt

```
1   package com.presca.modul2
2
3   import android.os.Bundle
4   import android.widget.Toast
5   import androidx.activity.ComponentActivity
6   import androidx.activity.compose.setContent
7   import androidx.compose.foundation.background
8   import androidx.compose.foundation.layout.*
9   import androidx.compose.foundation.rememberScrollState
10  import androidx.compose.foundation.selection.selectable
11  import
    androidx.compose.foundation.shape.RoundedCornerShape
12  import androidx.compose.foundation.text.KeyboardOptions
13  import androidx.compose.foundation.verticalScroll
14  import androidx.compose.material3.*
15  import androidx.compose.runtime.*
```

```
16  import
    androidx.compose.runtime.saveable.rememberSaveable
17  import androidx.compose.ui.Alignment
18  import androidx.compose.ui.Modifier
19  import androidx.compose.ui.graphics.Color
20  import androidx.compose.ui.platform.LocalContext
21  import androidx.compose.ui.text.input.KeyboardType
22  import androidx.compose.ui.text.style.TextAlign
23  import androidx.compose.ui.unit.dp
24  import androidx.compose.ui.unit.sp
25  import com.presca.modul2.ui.theme.Modul2Theme
26
27  class MainActivity : ComponentActivity() {
28      override fun onCreate(savedInstanceState: Bundle?)
    {
29          super.onCreate(savedInstanceState)
30          setContent {
31              Modul2Theme {
32                  Surface(
33                      modifier = Modifier.fillMaxSize(),
34                      color =
    MaterialTheme.colorScheme.background
35                  ) {
36                      TipCalculatorScreen()
37                  }
38              }
39          }
40      }
41  }
42
43  @Composable
44  fun TipCalculatorScreen() {
45      var costInput by rememberSaveable {
    mutableStateOf("") }
46      var selectedOption by rememberSaveable {
    mutableStateOf(20) }
47      var roundUp by rememberSaveable {
    mutableStateOf(false) }
48      var tipResult by rememberSaveable {
    mutableStateOf("Tip Amount") }
49
50      val context = LocalContext.current
51
52      fun calculateTip() {
53          val cost = costInput.toDoubleOrNull()
54
55          if (cost == null || cost <= 0) {
56              Toast.makeText(
57                  context,
```

```
58                  "Masukkan angka positif dan bukan
    nol!",
59                  Toast.LENGTH_SHORT
60              ).show()
61              return
62          }
63
64          val tip = TipCalculator.calculateTip(
65              cost = cost,
66              percentage = selectedOption,
67              roundUp = roundUp
68          )
69          tipResult = "Tip Amount:
    \$${"%.2f".format(tip)}"
70      }
71
72      Column(
73          modifier = Modifier
74              .verticalScroll(rememberScrollState())
75              .fillMaxWidth()
76      ) {
77
78          Box(
79              modifier = Modifier
80                  .fillMaxWidth()
81                  .background(Color(0xFF6200EE))
82                  .padding(16.dp)
83          ) {
84              Text(
85                  text = "Tip Time",
86                  color = Color.White,
87                  fontSize = 20.sp,
88                  modifier =
    Modifier.align(Alignment.CenterStart)
89              )
90          }
91
92          Column(
93              modifier = Modifier
94                  .padding(16.dp)
95                  .fillMaxWidth(),
96              verticalArrangement =
    Arrangement.spacedBy(16.dp)
97          ) {
98              OutlinedTextField(
99                  value = costInput,
100                 onValueChange = { costInput = it },
101                 label = { Text("Cost of Service") },
102                 modifier = Modifier.fillMaxWidth(),
```

```
103                 keyboardOptions =
    KeyboardOptions(keyboardType = KeyboardType.Number),
104                 shape = RoundedCornerShape(8.dp),
105                 colors = TextFieldDefaults.colors(
106                     focusedIndicatorColor =
    Color(0xFF6200EE),
107                     unfocusedIndicatorColor =
    Color.Gray,
108                     focusedContainerColor =
    Color.White,
109                     unfocusedContainerColor =
    Color.White,
110                     focusedLabelColor =
    Color(0xFF6200EE),
111                     unfocusedLabelColor = Color.Gray
112                 ),
113                 singleLine = true
114             )
115
116             Text(
117                 text = "How was the service?",
118                 color = Color.Gray
119             )
120
121             val options = listOf(
122                 "Amazing (20%)" to 20,
123                 "Good (18%)" to 18,
124                 "Okay (15%)" to 15
125             )
126
127             options.forEach { (label, value) ->
128                 Row(
129                     verticalAlignment =
    Alignment.CenterVertically,
130                     modifier = Modifier
131                         .fillMaxWidth()
132                         .selectable(
133                             selected = (selectedOption
    == value),
134                             onClick = { selectedOption
    = value }
135                         )
136                         .padding(4.dp)
137                 ) {
138                     RadioButton(
139                         selected = (selectedOption ==
    value),
140                         onClick = { selectedOption =
    value }
```

```
141                     )
142                     Text(text = label)
143                 }
144             }
145
146             Row(
147                 verticalAlignment =
    Alignment.CenterVertically
148             ) {
149                 Text("Round up tip?")
150                 Spacer(modifier = Modifier.weight(1f))
151                 Switch(
152                     checked = roundUp,
153                     onCheckedChange = { roundUp = it }
154                 )
155             }
156
157             Button(
158                 onClick = { calculateTip() },
159                 modifier = Modifier
160                     .fillMaxWidth()
161                     .height(48.dp),
162                 colors = ButtonDefaults.buttonColors(
163                     containerColor = Color(0xFF6200EE)
164                 ),
165                 shape = RoundedCornerShape(0.dp)
166             ) {
167                 Text("CALCULATE", color = Color.White,
    fontSize = 16.sp)
168             }
169
170             Text(
171                 text = tipResult,
172                 modifier = Modifier
173                     .fillMaxWidth()
174                     .padding(top = 4.dp),
175                 textAlign = TextAlign.End,
176                 color = Color.Gray
177             )
178         }
179     }
180 }
```

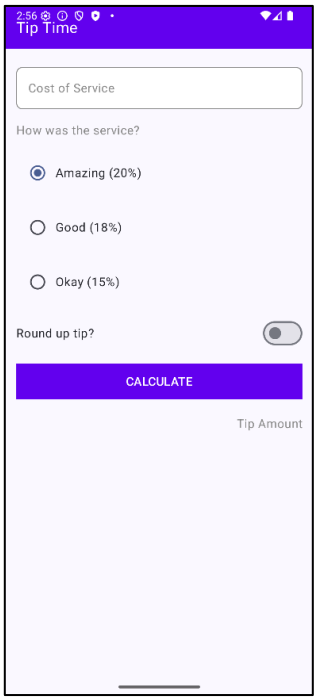
Tabel 3. Source Code Jawaban Soal 1 MainActivity.kt

2. TipCalculator.kt

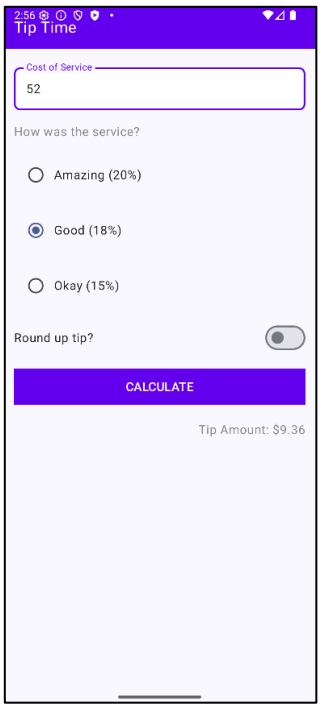
```
1   package com.presca.modul2
2
3   import kotlin.math.ceil
4
5   object TipCalculator {
6       fun calculateTip(cost: Double, percentage: Int,
    roundUp: Boolean): Double {
7           var tip = cost * percentage / 100
8           if (roundUp) tip = ceil(tip)
9           return tip
10      }
11  }
```

Tabel 4. Source Code Jawaban Soal 1 TipCalculator.kt

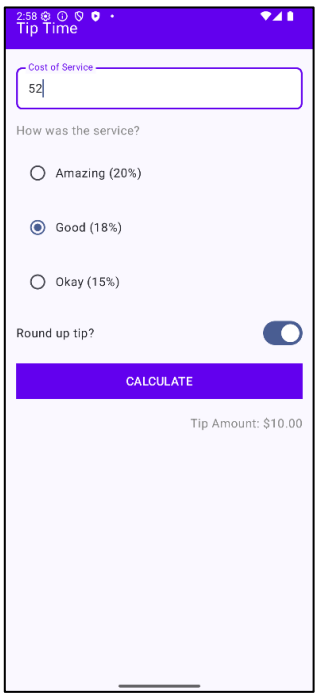
B. Output Program



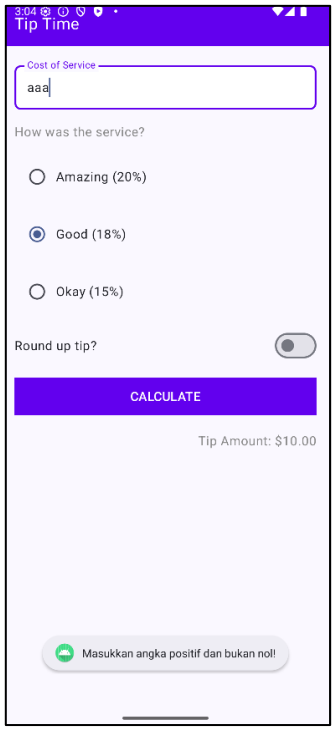
Gambar 10. Screenshot Hasil Jawaban Soal 1 Tampilan Awal



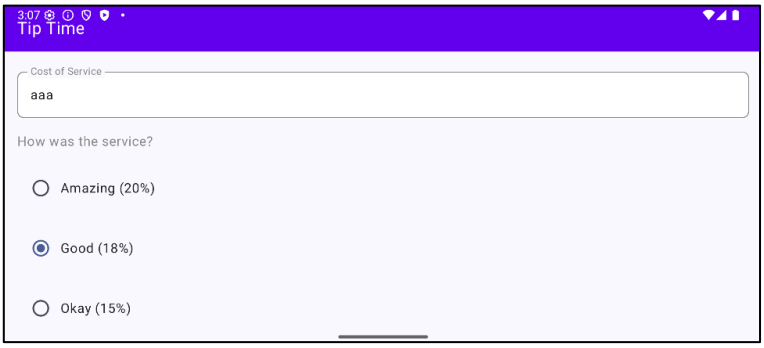
Gambar 11. Screenshot Hasil Jawaban Soal 1 Saat Tip Dihitung



Gambar 12. Screenshot Hasil Jawaban Soal 1 Saat Tip Dibulat



Gambar 13. Screenshot Hasil Jawaban Soal 1 Toast



Gambar 14. Screenshot Hasil Jawaban Soal 1 Mode Landscape

C. Pembahasan

3. MainActivity.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul2`.
- Pada baris [3] hingga [25], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [27], `class MainActivity : ComponentActivity()` ini digunakan sebagai titik awal aplikasi yang akan mengatur tampilan aplikasi.
- Pada baris [28], `onCreate()` merupakan siklus hidup (lifecycle) dari sebuah activity, yang dimana fungsi ini dipanggil pertama kali saat Activity dibuat.
- Pada baris [29], `super.onCreate(savedInstanceState)` ini memanggil `onCreate()` agar proses inisialisasi standar tetap berjalan.
- Pada baris [30], `setContent{…}` digunakan untuk menetapkan tampilan UI dari aplikasi, dan diterapkan tampilan dari `Prakmodul1Theme`.
- Pada baris [31], diterapkan tema khusus aplikasi dari `Modul2Theme`.
- Pada baris [32], `Surface` digunakan untuk wadah yang menampung UI.
- Pada baris [36], `TipCalculatorScreen()` dipanggil untuk menampilkan antarmuka kalkulator tip.
- Pada baris [43], @Composable merupakan anotasi dari Compose, yang dimana ini sebagai penanda bahwa `TipCalculatorScreen` adalah fungsi yang bisa digunakan dalam UI deklaratif Jetpack Compose.
- Pada baris [44], `fun TipCalculatorScreen()` bagian ini bisa digunakan untuk menampilkan UI, mengatur input, dan berinteraksi dengan state.
- Pada baris [50], `val context` ini akan mengambil Contect dari sistem Android yang digunakan untuk kebutuhan non-UI.

- Pada baris [52] dan [53], pada `calculateTip()` ini akan mengunggah costInput (string input teks) menjadi Double, jika gagal maka input tidak valid (bukan angka positif).
- Pada baris [55] hingga [61], dilakukan pengkondisian `if`, divalidasikan jika input yang dimasukkan pengguna kosong (null) atau kurang dari nol, maka akan muncul pesan "Masukkan angka positif dan bukan nol!"
- Pada baris [64] hingga [67], `val tip = TipCalculator.calculateTip()` ini akan memangil fungsi `TipCalculator.calculateTip` untuk menghitung nilai tip berdasarkan input biaya, persentase tip, dan pembulatannya.
- Pada baris [69], `tipResult` akan menampilkan hasil dari perhitungan tip dan akan `{"%.2f".format(tip)}` ini akan memberi format dua angka desimal di belakang koma.
- Pada baris [72], `Column` ini untuk menyusun komponen UI secara vertikal.
- Pada baris [74], `verticalScroll()` digunakan agar kolom pada bagian ini bisa discroll jika kontennya melebihi tinggi layar.
- Pada baris [75], `fillMaxWidth()` digunakan agar kolom memenuhi lebar layar.
- Pada baris [78] hingga [82], `Box` pada bagian ini digunakan untuk menyusun teks agar bisa rata kiri.
- Pada baris [84] hingga [88], `Text` ini digunakan untuk menampilkan teks "Tip Time" pada box.
- Pada baris [92] hingga [96], `Column` pada bagian ini digunakan untuk memberi jarak dari tepi layar (`padding`), digunakan `fillMaxWidth()` agar lebarnya memenuhi layar, dan `Arrangement.spacedBy(16.dp)` untuk memberikan jarak 16dp antar komponen di dalam kolom.

- Pada baris [98] hingga [113], `OutlinedTextField` ini untuk mengatur input teks dalam biaya layanan. `onValueChange = { costInput = it }` berfungsi untuk memperbarui `costInput` saat user mengetik, `keyboardOptions` digunakan agar input ini hanya menerima input angka (`KeyboardType.Number`), shape digunakan untuk membulatkan sudut border, `colors` untuk menyesuaikan warna, dan `singleLine = true` digunakan agar input tetap 1 baris.
- Pada baris [116] hingga [118], `Text` pada baris ini digunakan untuk menampilkan dan mengatur teks "How was the service?".
- Pada baris [121] hingga [124], dibuat radio button untuk memilihi persentase tip.
- Pada baris [127], dibuat iterasi `forEach` pada `options.forEach` untuk opsi di dalamnya.
- Pada baris [128] hingga [136], `Row` pada bagian ini untuk isi (RadioButton dan Teks) dalam bentuk horizontal. `selected = (selectedOption == value)` digunakan untuk menentukan apakah baris ini dipilih, `onClick = { selectedOption = value }` digunakan pada saat radio button diklik, ubah pilihan ke value yang sesuai.
- Pada baris [138] hingga [142], dibuat fungsi dari RadioButton dan labelnya. Pada saat radio button diklik, maka radio button akan tercentang (fungsi dari `selected = (selectedOption == value)`) dan `onClick = { selectedOption = value }` digunakan pada saat radio diklik, ubah nilai `selectedOption`. `Text` digunakan untuk menampilkan teks dari pilihan.
- Pada baris [146], `Row` ini untuk membuat baris horizontal yang berisi teks dan switch.
- Pada baris [149], `Text("Round up tip?")` ini untuk menampilkan teks Round up tip?".

- Pada baris [150], `Spacer(modifier = Modifier.weight(1f))` ini digunakan agar switch round up ada di bagian paling kanan.
- Pada baris [151] hingga [153], `Switch` ini adalah tombol on/off, jika switch statusnya on maka `roundUp` bernilai `true`.
- Pada baris [157] hingga [167], `Button` ini digunakan untuk mengatur tombol utama dari "Calculate". Ketika tombol ini ditekan, maka fungsi `calculateTip()` akan dipanggil untuk menghitung tip.
- Pada baris [170] hingga [176], `Text` pada bagian ini digunakan untuk menampilkan dan mengatur style hasil dari perhitungan tip.

4. TipCalculator.kt

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul2`.
- Pada baris [2], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya. Import pada baris ini akan mengimpor fungsi `ceil()` yang digunakan untuk membulatkan angka ke atas.
- Pada baris [5], `object` ini menandakan bahwa aka nada satu instance dari TipCalculator.
- Pada baris [6], `cost: Double` digunakan untuk nilai biaya layanan yang sudah dipastikan valid dan juga dikonversi dari input pengguna, `percentage: Int` untuk mengatur persentase tip yang dipilih, dan `roundup: Boolean` digunakan untuk memastikan apakah hasil tip perlu dibulatkan ke atas atau tidak.
- Pada baris [7], `var tip = cost * percentage / 100` ini digunakan untuk menghitung jumlah tip awal sebelum dibulatkan.

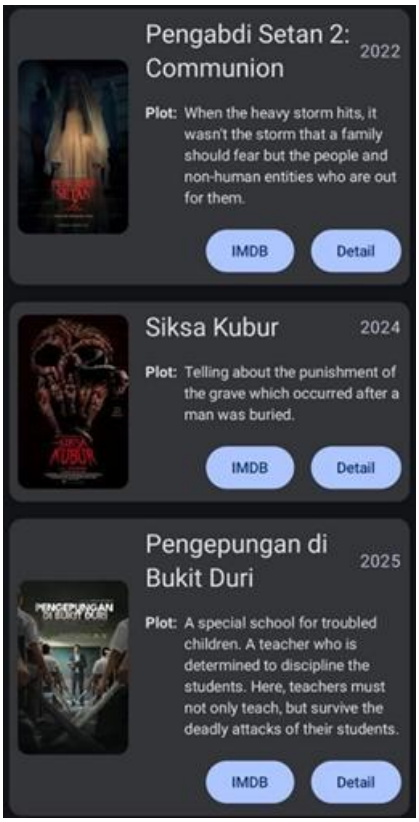
- Pada baris [8], `if (roundUp) tip = ceil(tip)` ini merupakan pengkondisian jika pengguna ingin membulatkan tip, maka tip akan dibulatkan ke atas.
- Pada baris [9], `return tip` ini untuk mengembalikan hasil perhitungan tip ke pemanggil fungsinya.

MODUL 3 : Build a Scrollable List

SOAL 1

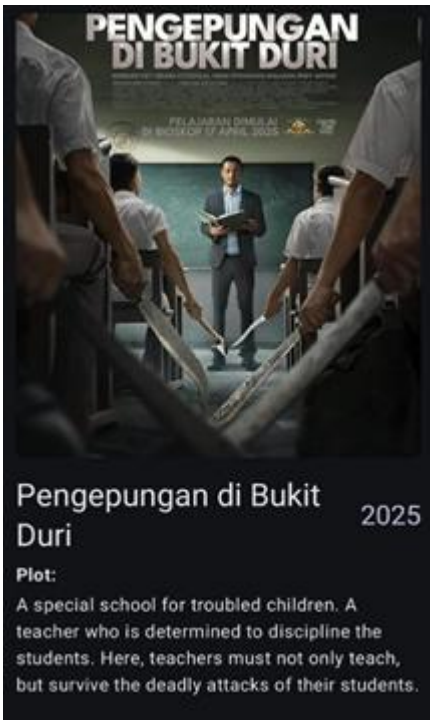
1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
    1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
    3. item pada 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
    5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
    6. 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
    7. Aplikasi menggunakan arsitektur single activity (satu activity memiliki beberapa fragment)
    8. Aplikasi berbasis XML harus menggunakan ViewBinding

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 15. Tampilan List Aplikasi

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 16. Tampilan Detail Aplikasi

A. Source Code

1. MainActivity.kt

```
1   package com.presca.modul3
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import androidx.activity.ComponentActivity
7   import androidx.activity.compose.setContent
8   import androidx.activity.enableEdgeToEdge
9   import androidx.compose.runtime.Composable
10  import androidx.compose.ui.platform.LocalContext
11  import androidx.navigation.NavHostController
12  import androidx.navigation.compose.NavHost
13  import androidx.navigation.compose.composable
14  import androidx.navigation.compose.rememberNavController
15  import com.presca.modul3.ui.theme.Modul3Theme
16
17  class MainActivity : ComponentActivity() {
```

```
18      override fun onCreate(savedInstanceState: Bundle?) {
19          super.onCreate(savedInstanceState)
20          enableEdgeToEdge()
21          setContent {
22              Modul3Theme {
23                  val navController =
    rememberNavController()
24                  AppNavigation(navController)
25              }
26          }
27      }
28  }
29
30  @Composable
31  fun AppNavigation(navController: NavHostController) {
32      val context = LocalContext.current
33
34      NavHost(
            navController = navController,
35          startDestination = "music_list"
36      ) {
37          composable("music_list") {
38              MusicListScreen(
39                  onMusicClick = { music ->
40
    navController.navigate("detail/${music.id}")
41                  },
42                  onExternalClick = { url ->
43                      val intent =
    Intent(Intent.ACTION_VIEW, Uri.parse(url))
44                      context.startActivity(intent)
45                  }
46              )
47          }
48
49          composable("detail/{musicId}") { backStackEntry
    ->
50              val musicId =
    backStackEntry.arguments?.getString("musicId")?.toIntOrN
    ull()
                val music = musicList.find { it.id ==
    musicId }
51              music?.let {
52                  MusicDetailScreen(music = it,
    navController = navController)
53              }
54          }
55      }
56  }
```

```
58
59
```

Tabel 5. Source Code Jawaban Soal 1 MainActivity.kt

2. MusicData.kt

```
1  package com.presca.modul3
2
3  data class Music(
4      val id: Int,
5      val title: String,
6      val year: String,
7      val imageUrl: String,
8      val externalUrl: String,
9      val description: String
10 )
11
12 val musicList = listOf(
13     Music(
14         id = 1,
15         title = "What is Love?",
16         year = "9 April 2018",
17         imageUrl = "https://i.postimg.cc/5tpLc5DY/What-
   Is-Love-Online-Cover.webp",
18         externalUrl =
   "https://en.wikipedia.org/wiki/What_Is_Love%3F_(Twice_so
   ng)",
19         description = "Lagu \"What is Love?\"
   menceritakan tentang rasa penasaran seorang gadis muda
   yang belum pernah merasakan cinta sejati. Ia hanya
   mengetahui tentang cinta dari film, drama, dan buku,
   sehingga muncul keinginan kuat untuk benar-benar
   memahami dan merasakannya sendiri. Lirik lagu ini
   menggambarkan perasaan ingin tahu dan harapan akan
   datangnya cinta yang indah seperti yang sering
   digambarkan dalam kisah romantis di layar kaca."
20     ),
21     Music(
22         id = 2,
23         title = "Fancy",
24         year = "22 April 2019",
25         imageUrl =
   "https://upload.wikimedia.org/wikipedia/id/0/09/Twice_-
   _Fancy_You.png",
26         externalUrl =
   "https://twice.fandom.com/wiki/Fancy",
```

```
27          description = "\"FANCY\" bercerita tentang
    perasaan jatuh cinta yang begitu kuat, penuh keberanian,
    dan tidak takut mengambil risiko. Lagu ini menggambarkan
    momen ketika seseorang naksir atau menyukai orang lain
    dengan sangat intens, bahkan meski terasa berbahaya
    seperti duri mawar, perasaan itu tetap terasa manis dan
    menyenangkan. TWICE mengekspresikan keinginan untuk
    mengungkapkan cinta secara langsung, tanpa ragu, dan
    berani mengambil langkah pertama. Lirik seperti “Aku
    menyukaimu, aku menyukaimu, menyukaimu” dan “Pegang
    lebih kuat, ambil tanganku. Ini akan sedikit berbahaya,
    bahkan lebih berbahaya lagi, sayang” menegaskan
    keberanian dalam menghadapi cinta yang penuh tantangan."
28      ),
29      Music(
30          id = 3,
31          title = "The Feels",
32          year = "1 Oktober 2021",
33          imageUrl =
    "https://upload.wikimedia.org/wikipedia/en/5/50/Twice_-
    _The_Feels.png",
34          externalUrl =
    "https://twice.fandom.com/wiki/The_Feels",
35          description = "\"The Feels\" menggambarkan
    perasaan jatuh cinta yang kuat namun penuh rasa malu,
    seperti yang sering dialami remaja pada cinta pertama.
    Lagu ini bercerita tentang dua orang yang saling
    menyukai, namun keduanya masih malu-malu untuk
    mengungkapkan perasaan mereka secara langsung. TWICE
    mengekspresikan kegembiraan, rasa penasaran, dan degup
    jantung yang tak tertahankan saat jatuh cinta, sekaligus
    memberi kode kepada orang yang disukai agar lebih berani
    mengungkapkan perasaannya."
36      ),
37      Music(
38          id = 4,
39          title = "Likey",
40          year = "30 Oktober 2017",
41          imageUrl =
    "https://i.scdn.co/image/ab67616d0000b2731f21c24e81a9d0d
    4a30be533",
42          externalUrl =
    "https://twice.fandom.com/wiki/Likey",
43          description = "\"Likey\" menceritakan tentang
    perasaan suka dan jatuh cinta yang membuat seseorang
    merasa bersemangat sekaligus sedikit canggung. Lagu ini
    menggambarkan bagaimana sang tokoh utama membangun
    kepercayaan diri untuk mengungkapkan perasaannya kepada
    orang yang disukai, meski masih ada rasa malu dan ragu.
```

```
    Selain itu, lagu ini juga mengangkat tema tentang
    bagaimana seseorang ingin tampil menarik dan disukai,
    terutama di era media sosial, sehingga ada tekanan untuk
    selalu menunjukkan sisi terbaiknya agar mendapatkan
    \"like\" atau perhatian dari orang lain. Melalui
    liriknya, TWICE menyampaikan pesan tentang keberanian
    dalam mencintai dan menjadi diri sendiri, tanpa harus
    terlalu terpengaruh oleh penilaian orang lain."
44      ),
45      Music(
46          id = 5,
47          title = "Cheer Up",
48          year = "24 April 2016",
49          imageUrl =
    "https://i.scdn.co/image/ab67616d0000b273acf4830dde5e17d
    356b80ae8",
50          externalUrl =
    "https://en.wikipedia.org/wiki/Cheer_Up_(song)",
51          description = "Lagu ini mengisahkan tentang
    seorang perempuan yang sedang dalam tahap pendekatan
    dengan pria yang sangat posesif, di mana ia meminta
    sedikit ruang agar proses pendekatan menjadi lebih
    menyenangkan dan tidak menekan. Melalui liriknya,
    \"Cheer Up\" menyampaikan pesan untuk tetap semangat dan
    memberikan dukungan kepada orang yang disukai, meskipun
    ada rasa canggung dan ketidakpastian dalam hubungan yang
    sedang berkembang. Lagu ini juga menonjolkan karakter
    ceria dan energik khas TWICE, dengan melodi yang catchy
    dan koreografi yang dinamis."
52      ),
53      Music(
54          id = 6,
55          title = "TT",
56          year = "24 Oktober 2016",
57          imageUrl =
    "https://i.scdn.co/image/ab67616d0000b273387444ab2fc1f08
    dfe7915ab",
58          externalUrl =
    "https://en.wikipedia.org/wiki/TT_(song)",
59          description = "Lagu \"TT\" menggambarkan
    perasaan cinta yang membingungkan dan penuh gejolak.
    Liriknya menceritakan tentang seorang perempuan yang
    merasa kesal dan bingung terhadap perasaan cintanya yang
    semakin menggebu-gebu meskipun ia berusaha menjauh.
    Perasaan tersebut membuatnya merasa seperti \"TT\" -
    sebuah ekspresi emotikon menangis yang juga menjadi
    simbol khas lagu ini. Lagu ini mengungkapkan dilema
    antara ingin mendekat dan sekaligus merasa canggung atau
    takut dalam menghadapi cinta."
```

```
60      )
61  )
```

Tabel 6. Source Code Jawaban Soal 1 MusicData.kt

3. MusicListScreen.kt

```
1   package com.presca.modul3
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.lazy.LazyColumn
5   import androidx.compose.foundation.lazy.items
6   import
    androidx.compose.foundation.shape.RoundedCornerShape
7   import androidx.compose.material3.*
8   import androidx.compose.runtime.Composable
9   import androidx.compose.ui.Modifier
10  import androidx.compose.ui.draw.clip
11  import androidx.compose.ui.unit.dp
12  import androidx.compose.ui.unit.sp
13  import androidx.compose.ui.text.font.FontWeight
14  import
    com.bumptech.glide.integration.compose.ExperimentalGlid
    eComposeApi
15  import
    com.bumptech.glide.integration.compose.GlideImage
16
17  @OptIn(ExperimentalGlideComposeApi::class,
    ExperimentalMaterial3Api::class)
18  @Composable
19  fun MusicListScreen(
20      onMusicClick: (Music) -> Unit,
21      onExternalClick: (String) -> Unit
22  ) {
23      Scaffold(
24          topBar = {
25              TopAppBar(
26                  title = { Text("List of Twice's Best
    Songs") }
27              )
28          }
29      ) { padding ->
30          LazyColumn(
31              contentPadding = padding,
32              modifier = Modifier
33                  .fillMaxSize()
34                  .padding(8.dp)
```

```
35          ) {
36              items(musicList) { music ->
37                  Card(
38                      modifier = Modifier
39                          .fillMaxWidth()
40                          .padding(vertical = 10.dp),
41                      shape = RoundedCornerShape(16.dp),
42                      elevation =
    CardDefaults.cardElevation(6.dp)
43                  ) {
44                      Row(modifier =
    Modifier.padding(12.dp)) {
45                          GlideImage(
46                              model = music.imageUrl,
47                              contentDescription =
    music.title,
48                              modifier = Modifier
49                                  .size(120.dp)

50  .clip(RoundedCornerShape(12.dp))
51                          )
52                          Spacer(Modifier.width(16.dp))
53                          Column(modifier =
    Modifier.weight(1f)) {
54                              Text(
55                                  music.title,
56                                  fontSize = 18.sp,
57                                  style =
    MaterialTheme.typography.titleMedium
58                              )
59                              Text(
60                                  "Tanggal Rilis:
    ${music.year}",
61                                  fontSize = 14.sp,
62                                  color =
    MaterialTheme.colorScheme.onBackground.copy(alpha =
    0.6f)
63                              )

64  Spacer(Modifier.height(8.dp))
65                              Text(
66                                  text = "Tentang Lagu
    Ini:",
67                                  fontSize = 12.sp,
68                                  style =
    MaterialTheme.typography.bodySmall.copy(fontWeight =
    FontWeight.Bold)
69                              )
70                              Text(
```

```
71                                      text =
    music.description,
72                                      fontSize = 12.sp,
73                                      maxLines = 3,
74                                      style =
    MaterialTheme.typography.bodySmall
75                                  )
76
    Spacer(Modifier.height(12.dp))
77                                  Row(
78                                      modifier =
    Modifier.fillMaxWidth(),
79                                      horizontalArrangement =
    Arrangement.spacedBy(8.dp)
80                                  ) {
81                                      Button(
82                                          onClick = {
    onExternalClick(music.externalUrl) },
83                                          modifier =
    Modifier.weight(1f)
84                                      ) {
85                                          Text("Info")
86                                      }
87                                      Button(
88                                          onClick = {
    onMusicClick(music) },
89                                          modifier =
    Modifier.weight(1f)
90                                      ) {
91                                          Text("Detail")
92                                      }
93                                  }
94                              }
95                          }
96                      }
97                  }
98              }
99          }
100 }
```

Tabel 7. Source Code Jawaban Soal 1 MusicListScreen.kt

4. MusicDetailScreen.kt

```
1   package com.presca.modul3
2
3   import androidx.compose.foundation.layout.*
```

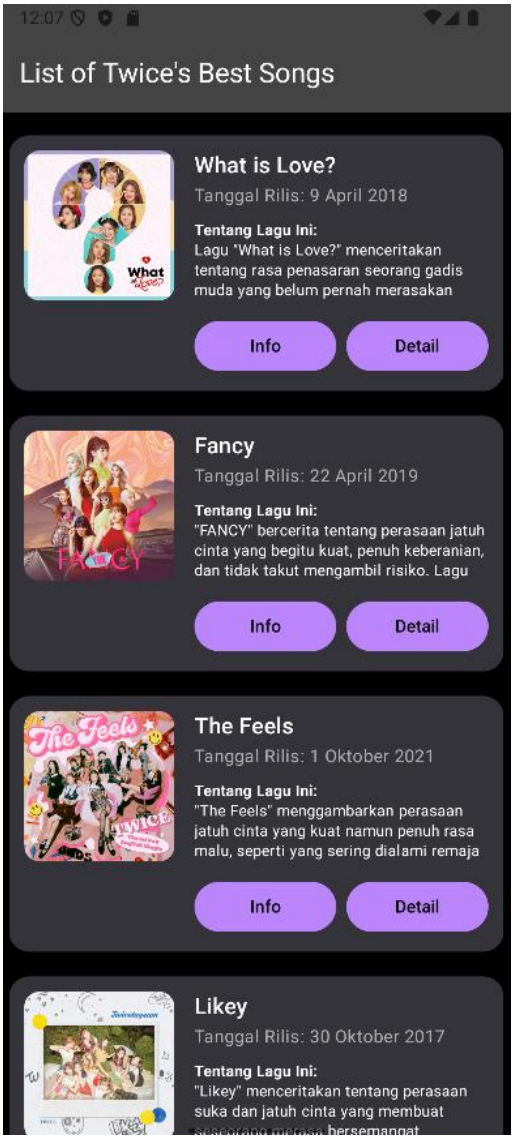
```
4   import androidx.compose.foundation.lazy.LazyColumn
5   import androidx.compose.foundation.lazy.items
6   import androidx.compose.material3.*
7   import androidx.compose.material.icons.Icons
8   import androidx.compose.material.icons.filled.ArrowBack
9   import androidx.compose.runtime.Composable
10  import androidx.compose.ui.Modifier
11  import androidx.compose.ui.unit.dp
12  import androidx.navigation.NavController
13  import
    com.bumptech.glide.integration.compose.ExperimentalGlid
    eComposeApi
14  import
    com.bumptech.glide.integration.compose.GlideImage
15  import
    androidx.compose.foundation.shape.RoundedCornerShape
16  import androidx.compose.ui.draw.clip
17
18  @OptIn(ExperimentalGlideComposeApi::class,
    ExperimentalMaterial3Api::class)
19  @Composable
20  fun MusicDetailScreen(music: Music, navController:
    NavController) {
21      Scaffold(
22          topBar = {
23              TopAppBar(
24                  title = {
25                      Text(
26                          text = music.title,
27                          color =
    MaterialTheme.colorScheme.onPrimary
28                      )
29                  },
30                  navigationIcon = {
31                      IconButton(
32                          onClick = {
    navController.popBackStack() },
33                          colors =
    IconButtonDefaults.iconButtonColors(
34                              contentColor =
    MaterialTheme.colorScheme.onPrimary
35                          )
36                      ) {
37                          Icon(
38                              imageVector =
    Icons.Filled.ArrowBack,
39                              contentDescription = "Back"
40                          )
41                      }
```

```
42                  },
43                  colors =
TopAppBarDefaults.topAppBarColors(
44                      containerColor =
MaterialTheme.colorScheme.surface
45                  )
46              )
47          },
48          containerColor =
MaterialTheme.colorScheme.background
49      ) { padding ->
50          LazyColumn(
51              modifier = Modifier
52                  .padding(padding)
53                  .padding(16.dp)
54                  .fillMaxSize(),
55              verticalArrangement =
Arrangement.spacedBy(12.dp)
56          ) {
57              item {
58                  GlideImage(
59                      model = music.imageUrl,
60                      contentDescription = music.title,
61                      modifier = Modifier
62                          .fillMaxWidth()
63                          .height(400.dp)

64  .clip(RoundedCornerShape(16.dp))
65                  )
66              }
67              item {
68                  Text("Judul: ${music.title}", style =
MaterialTheme.typography.titleLarge)
69              }
70              item {
71                  Text("Tanggal Rilis: ${music.year}")
72              }
73              item {
74                  Text("Tentang Lagu Ini:", style =
MaterialTheme.typography.titleMedium)
75              }
76              item {
77                  Text(
78                      text = music.description,
79                      style =
MaterialTheme.typography.bodyMedium
80                  )
81              }
82          }
```

```
83          }
84      }
85
```

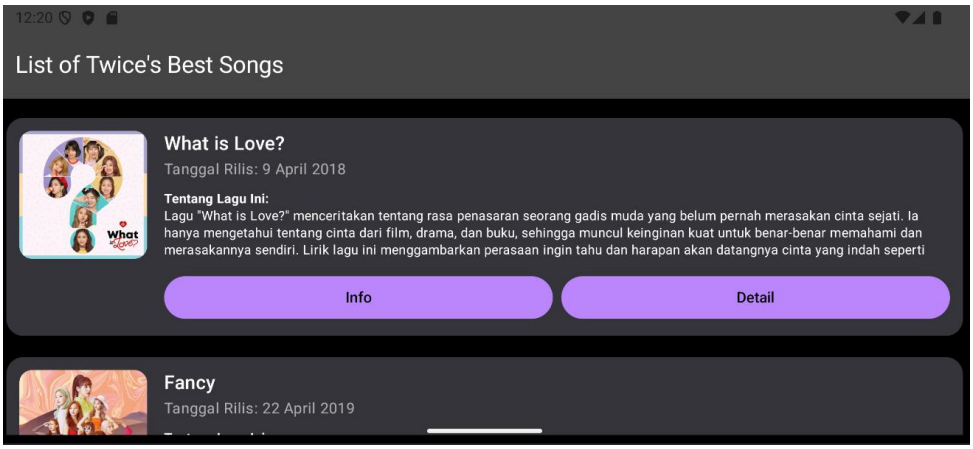
Tabel 8. Source Code Jawaban Soal 1 MusicDetailScreen.kt

B. Output Program



Gambar 17. Screenshot Hasil Jawaban Soal 1 Tampilan List

Gambar 18. Screenshot Hasil Jawaban Soal 1 Tampilan Detail



Gambar 19. Screenshot Hasil Jawaban Soal 1 Mode Landscape

C. Pembahasan

1) Berikut adalah penjelasan untuk soal nomor 1:

1. MainActivity.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul3`.
- Pada baris [3] hingga [15], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [17], `class MainActivity : ComponentActivity()` ini digunakan sebagai titik awal aplikasi yang akan mengatur tampilan aplikasi.
- Pada baris [18], `onCreate()` merupakan siklus hidup (lifecycle) dari sebuah activity, yang dimana fungsi ini dipanggil pertama kali saat Activity dibuat.
- Pada baris [19], `super.onCreate(savedInstanceState)` ini memanggil `onCreate()` agar proses inisialisasi standar tetap berjalan.
- Pada baris [20], `enableEdgeToEdge()` digunakan untuk mengaktifkan rendering hingga ke edge layar.

- Pada baris [21], `setContent{…}` digunakan untuk menetapkan tampilan UI dari aplikasi, dan diterapkan tampilan dari `Modul3Theme`.
- Pada baris [22], diterapkan tema khusus aplikasi dari `Modul3Theme`.
- Pada baris [23], `rememberNavController()` digunakan untuk membuat controller navigasi yang mengatur perpindahan antar-screen.
- Pada baris [24], `AppNavigation(navController)` berisi logika navigasi aplikasi.
- Pada baris [31], `AppNavigation` adalah fungsi compose yang digunakan untuk menangani semua screen dan navigasi antar-screen.
- Pada baris [32], `LocalContext.current` ini berisi akses Context Android dari dalam Compose.
- Pada baris [34], `NavHost` merupakan container dari navigasi.
- Pada baris [36], `startDestination` ini digunakan untuk mengatur screen awal adalah "music_list".
- Pada baris [38], didefinisikan “music_list” untuk menampilkan daftar musik.
- Pada baris [39], `MusicListScreen(...)` merupakan fungsi `@Composable` yang bertanggung jawab menampilkan daftar musik.
- Pada baris [40] dan [41], fungsi `onMusicClick` ini akan terpanggil ketika pengguna menekan button “Detail” dan akan menampilkan detail musik.
- Pada baris [43] hingga [45], fungsi `onExternalClick` ini akan terpanggil ketika pengguna menekan button “Info” dan halaman akan berpindah membuka URL di browser.
- Pada baris [50], `composable("detail/{musicId}")` ini digunakan untuk mendefinisikan route untuk tampilan detail musik.

- Pada baris [51], `val music = musicList.find { it.id == musicId }` ini akan mencari music yang sesuai dari `musicList`, yang dimana isi list ini adalah berisi data lagu.
- Pada baris [52], Jika music ditemukan (tidak null), maka tampilkan `MusicDetailScreen`. Informasi ini ditampilkan sesuai dengan ID.

2. MusicData.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul3`.
- Pada baris [3], didefinisikan data dengan nama `Music`.
- Pada baris [4], didefinisikan juga `id` dalam bentuk integer.
- Pada baris [5] hingga [9], didefinisikan `title`, `year`, `imageUrl`, `externalUrl`, dan `description` dalam bentuk string.
- Pada baris [12], `musicList` ini digunakan untuk membuat list dari musik yang nantinya akan ditampilkan pada `MusicListScreen`.
- Pada baris [13] hingga [19], diisi dari list musik pertama dengan format `id` (sesuai urutan lagu), `title` (judul lagu), `year` (tanggal rilis), `imageUrl` (gambar album), `externalUrl` (link yang dapat dibuka untuk informasi lebih jelas di browser, dan `description` (untuk penjelasan lebih detailnya).
- Pada baris [21] hingga [59], sesuaikan struktur sebelumnya untuk mengisi sisa lagu yang ingin dibuat ke dalam list.

3. MusicDetailScreen.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul3`.

- Pada baris [3] hingga [16], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [18], `@OptIn(…)` ini digunakan untuk mengindikasikan jika kita menggunakan API eksperimental (Glide Compose dan Material3).
- Pada baris [20], `MusicDetailScreen` dibuat dengan parameter `music` untuk data musik, dan `navController` untuk navigasi antar screen.
- Pada baris [21], `Scaffold` digunakan untuk membuat struktur layout utama.
- Pada baris [22], `topBar` digunakan untuk mendefinisikan bar di bagian atas.
- Pada baris [24], `title` nantinya pada bagian ini digunakan untuk menampilkan judul lagu pada bar atas, dan `text` nantinya digunakan untuk menampilkan teks pada judul.
- Pada baris [30], bagian `navigationIcon` ini digunakan untuk mengatur tombol navigasi (seperti tombol kembali).
- Pada baris [32], `onClick` pada bagian ini, `navController.popBackStack()` digunakan untuk kembali ke screen sebelumnya dalam stack navigasi
- Pada baris [38], pada bagian `Icon`, `imageVector = Icons.Filled.ArrowBack:` ini artinya kita menggunakan ikon panah kembali sebagai icon dari tombol kembali.
- Pada baris [43] dan [44], `colors` pada bagian ini digunakan untuk mengatur warna pada top bar.
- Pada baris [48], `containerColor` digunakan untuk mengatur warna latar belakang pada scaffold.
- Pada baris [50], `LazyColumn` merupakan sebuah composable yang digunakan untuk menampilkan daftar item yang dapat digulir secara vertikal.

- Pada baris [51] hingga [54], `modifier` pada bagian ini digunakan untuk mengatur layout dari `LazyColumn`.
- Pada baris [55], `Arrangement.spacedBy(12.dp)` ini digunakan untuk memberi jarak 12dp secara vertikal antar item.
- Pada baris [57], `item` pada bagian ini digunakan untuk mengatur ukuran dan bentuk dari cover musik.
- Pada baris [58], `GlideImage` ini merupakan komponen yang digunakan untuk menampilkan gambar dari URL menggunakan library Glide.
- Pada baris [67] hingga [79], `item` pada bagian tersebut mengatur kembali untuk tampilan teks dari judul, tanggal rilis, tentang lagu ini, dan teks deskripsi.

4. MusicListScreen.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul3`.
- Pada baris [3] hingga [15], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [18], `@OptIn(…)` ini digunakan untuk mengindikasikan jika kita menggunakan API eksperimental (Glide Compose dan Material3).
- Pada baris [19] hingga [21], `MusicListScreen` dibuat dengan parameter callback `onMusicClick` dan `onExternalClick` untuk interaksi pengguna.
- Pada baris [23], `Scaffold` digunakan untuk membuat struktur layout utama.
- Pada baris [24], `topBar` digunakan untuk mendefinisikan bar di bagian atas.

- Pada baris [26], `title` nantinya pada bagian ini digunakan untuk menampilkan judul lagu pada bar atas, dan `text` nantinya digunakan untuk menampilkan teks pada bar atas.
- Pada baris [30], `LazyColumn` merupakan sebuah composable yang digunakan untuk menampilkan daftar item yang dapat digulir secara vertikal.
- Pada baris [32] hingga [34], `modifier` pada bagian ini digunakan untuk mengatur layout dari `LazyColumn`.
- Pada baris [36], `items(musicList)` ini merupakan loop melalui daftar musik untuk membuat item.
- Pada baris [37], Card digunakan sebagai komponen Material Design yang digunakan untuk menampilkan konten terkelompok.
- Pada baris [39], `fillMaxWidth()` ini digunakan agar lebar mengisi parent.
- Pada baris [41], `RoundedCornerShape(16.dp)` ini digunakan untuk mengatur sudut kelengkungan dengan radius 16dp
- Pada baris [42], `cardElevation(6.dp)` digunakan untuk mengatur efek bayangan dengan elevasi 6dp.
- Pada baris [44], `Row` digunakan untuk layout horizontal untuk gambar dan juga teks informasi.
- Pada baris [45], `GlideImage` digunakan untuk menampilkan gambar cover dari URL.
- Pada baris [54], `Text` pada bagian ini akan mengatur tampilan dari judul.
- Pada baris [65] hingga [74], `Text` pada bagian tersebut mengatur kembali untuk tampilan teks dari tanggal rilis, tentang lagu ini, dan teks deskripsi.
- Pada baris [77] hingga [91], bagian ini digunakan untuk mengatur tampilan dari `button` "Info" dan "Detail" serta fungsi dari button tersebut pada saat diklik.

2) RecyclerView masih banyak digunakan karena RecyclerView stabil, fleksibel, dan juga mempunyai ekosistem library yang matang. Meskipun penulisan kodenya yang panjang dan bersifat boiler-plate, RecyclerView mendukung berbagai fitur lanjutan seperti multiple view types, custom item decoration, dan juga animasi kompleks. Selain itu, banyak aplikasi lama yang masih menggunakan XML sehingga tetap mengandalkan RecyclerView untuk mempertahankan kompatibilitas dan performa. Lazycolumn belum sepenuhnya menggantikan fleksibilitas RecyclerView dalam beberapa kasus yang spesifik.

Kelebihan RecyclerView dibanding LazyColumn:

- Stabil dan digunakan secara luas dalam produksi.
- Mendukung multiple view types dan pengaturan layout yang kompleks.
- Kompatibel dengan pendekatan UI berbasis XML yang masih digunakan secara luas.
- Banyak library pihak ketiga yang sudah terintegrasi dengan RecyclerView.
- Performa sangat baik dalam menangani daftar data besar dengan efisiensi tinggi.

SOAL 2

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

A. Pembahasan

RecyclerView masih banyak digunakan karena RecyclerView stabil, fleksibel, dan juga mempunyai ekosistem library yang matang. Meskipun penulisan kodenya yang panjang dan bersifat boiler-plate, RecyclerView mendukung berbagai fitur lanjutan seperti multiple view types, custom item decoration, dan juga animasi kompleks. Selain itu, banyak aplikasi lama yang masih menggunakan XML sehingga tetap mengandalkan RecyclerView untuk mempertahankan kompatibilitas dan performa. Lazycolumn belum sepenuhnya menggantikan fleksibilitas RecyclerView dalam beberapa kasus yang spesifik.

Kelebihan RecyclerView dibanding LazyColumn:

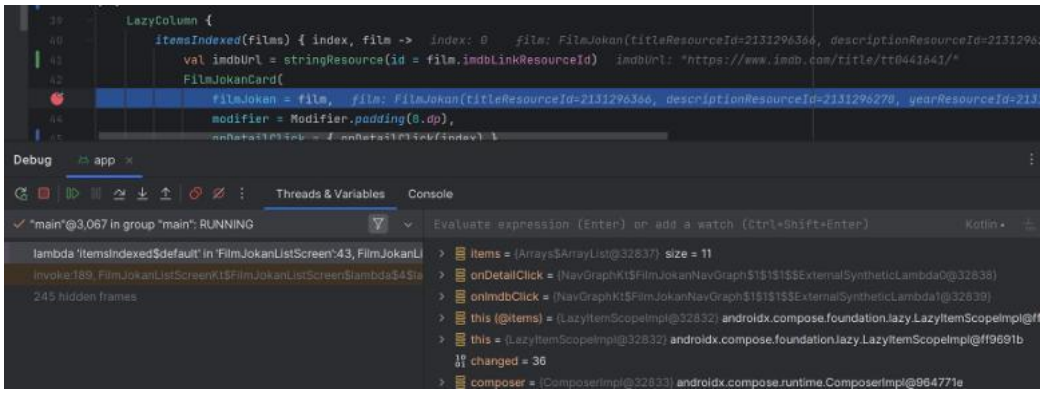
- Stabil dan digunakan secara luas dalam produksi.
- Mendukung multiple view types dan pengaturan layout yang kompleks.
- Kompatibel dengan pendekatan UI berbasis XML yang masih digunakan secara luas.
- Banyak library pihak ketiga yang sudah terintegrasi dengan RecyclerView.
- Performa sangat baik dalam menangani daftar data besar dengan efisiensi tinggi.

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 dalam pembuatan ViewModel
   c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
   d. gunakan logging untuk 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

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



Gambar 20. Contoh Penggunaan Debugger

A. Source Code

1. MainActivity.kt

```
1   package com.presca.modul4
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import androidx.activity.ComponentActivity
7   import androidx.activity.compose.setContent
8   import androidx.activity.enableEdgeToEdge
9   import androidx.activity.viewModels
10  import androidx.compose.runtime.Composable
11  import androidx.compose.ui.platform.LocalContext
12  import androidx.navigation.NavHostController
13  import androidx.navigation.compose.NavHost
14  import androidx.navigation.compose.composable
15  import androidx.navigation.compose.rememberNavController
16  import com.presca.modul4.ui.screens.MusicDetailScreen
17  import com.presca.modul4.ui.screens.MusicListScreen
18  import com.presca.modul4.ui.theme.Modul4Theme
19  import com.presca.modul4.viewmodel.MusicViewModel
20  import com.presca.modul4.viewmodel.MusicViewModelFactory
21
22  class MainActivity : ComponentActivity() {
23      private val viewModel: MusicViewModel by viewModels
    { MusicViewModelFactory() }
24
25      override fun onCreate(savedInstanceState: Bundle?) {
26          super.onCreate(savedInstanceState)
27          enableEdgeToEdge()
28          setContent {
29              Modul4Theme {
30                  val navController =
```

```
    rememberNavController()
31                  AppNavigation(navController, viewModel)
32              }
33          }
34      }
35  }
36
37  @Composable
38  fun AppNavigation(navController: NavHostController,
    viewModel: MusicViewModel) {
39      val context = LocalContext.current
40      NavHost(navController, startDestination =
    "music_list") {
41          composable("music_list") {
42              MusicListScreen(
43                  viewModel = viewModel,
44                  onMusicClick = {
45                      viewModel.logDetailClick()
46                      viewModel.logSelect(it)

47  navController.navigate("detail/${it.id}")
48                  },
49                  onExternalClick = {
50                      viewModel.logExternalClick(it)

51  context.startActivity(Intent(Intent.ACTION_VIEW,
    Uri.parse(it)))
52                  }
53              )
54          }
55          composable("detail/{musicId}") { backStackEntry
    ->
56              val musicId =
    backStackEntry.arguments?.getString("musicId")?.toIntOrN
    ull()
57              val music = viewModel.musicList.value.find {
    it.id == musicId }
58              music?.let {
59                  MusicDetailScreen(it, navController)
60              }
61          }
62      }
63  }
```

Tabel 9. Source Code Jawaban Soal 1 MainActivity.kt

2. MusicData.kt

```
1  package com.presca.modul4.data
2
3  import com.presca.modul4.models.Music
4
5  val twiceMusicList = listOf(
6      Music(
7          id = 1,
8          title = "What is Love?",
9          year = "9 April 2018",
10         imageUrl = "https://i.postimg.cc/5tpLc5DY/What-
   Is-Love-Online-Cover.webp",
11         externalUrl =
   "https://en.wikipedia.org/wiki/What_Is_Love%3F_(Twice_so
   ng)",
12         description = "Lagu \"What is Love?\"
   menceritakan tentang rasa penasaran seorang gadis muda
   yang belum pernah merasakan cinta sejati. Ia hanya
   mengetahui tentang cinta dari film, drama, dan buku,
   sehingga muncul keinginan kuat untuk benar-benar
   memahami dan merasakannya sendiri. Lirik lagu ini
   menggambarkan perasaan ingin tahu dan harapan akan
   datangnya cinta yang indah seperti yang sering
   digambarkan dalam kisah romantis di layar kaca."
13     ),
14     Music(
15         id = 2,
16         title = "Fancy",
17         year = "22 April 2019",
18         imageUrl =
   "https://upload.wikimedia.org/wikipedia/id/0/09/Twice_-
   _Fancy_You.png",
19         externalUrl =
   "https://twice.fandom.com/wiki/Fancy",
20         description = "\"FANCY\" bercerita tentang
   perasaan jatuh cinta yang begitu kuat, penuh keberanian,
   dan tidak takut mengambil risiko. Lagu ini menggambarkan
   momen ketika seseorang naksir atau menyukai orang lain
   dengan sangat intens, bahkan meski terasa berbahaya
   seperti duri mawar, perasaan itu tetap terasa manis dan
   menyenangkan. TWICE mengekspresikan keinginan untuk
   mengungkapkan cinta secara langsung, tanpa ragu, dan
   berani mengambil langkah pertama. Lirik seperti “Aku
   menyukaimu, aku menyukaimu, menyukaimu” dan “Pegang
   lebih kuat, ambil tanganku. Ini akan sedikit berbahaya,
   bahkan lebih berbahaya lagi, sayang” menegaskan
   keberanian dalam menghadapi cinta yang penuh tantangan."
21     ),
22     Music(
23         id = 3,
```

```
24          title = "The Feels",
25          year = "1 Oktober 2021",
26          imageUrl =
"https://upload.wikimedia.org/wikipedia/en/5/50/Twice_-
_The_Feels.png",
27          externalUrl =
"https://twice.fandom.com/wiki/The_Feels",
28          description = "\"The Feels\" menggambarkan
perasaan jatuh cinta yang kuat namun penuh rasa malu,
seperti yang sering dialami remaja pada cinta pertama.
Lagu ini bercerita tentang dua orang yang saling
menyukai, namun keduanya masih malu-malu untuk
mengungkapkan perasaan mereka secara langsung. TWICE
mengekspresikan kegembiraan, rasa penasaran, dan degup
jantung yang tak tertahankan saat jatuh cinta, sekaligus
memberi kode kepada orang yang disukai agar lebih berani
mengungkapkan perasaannya."
29      ),
30      Music(
31          id = 4,
32          title = "Likey",
33          year = "30 Oktober 2017",
34          imageUrl =
"https://i.scdn.co/image/ab67616d0000b2731f21c24e81a9d0d
4a30be533",
35          externalUrl =
"https://twice.fandom.com/wiki/Likey",
36          description = "\"Likey\" menceritakan tentang
perasaan suka dan jatuh cinta yang membuat seseorang
merasa bersemangat sekaligus sedikit canggung. Lagu ini
menggambarkan bagaimana sang tokoh utama membangun
kepercayaan diri untuk mengungkapkan perasaannya kepada
orang yang disukai, meski masih ada rasa malu dan ragu.
Selain itu, lagu ini juga mengangkat tema tentang
bagaimana seseorang ingin tampil menarik dan disukai,
terutama di era media sosial, sehingga ada tekanan untuk
selalu menunjukkan sisi terbaiknya agar mendapatkan
\"like\" atau perhatian dari orang lain. Melalui
liriknya, TWICE menyampaikan pesan tentang keberanian
dalam mencintai dan menjadi diri sendiri, tanpa harus
terlalu terpengaruh oleh penilaian orang lain."
37      ),
38      Music(
39          id = 5,
40          title = "Cheer Up",
41          year = "24 April 2016",
42          imageUrl =
"https://i.scdn.co/image/ab67616d0000b273acf4830dde5e17d
356b80ae8",
```

```
43          externalUrl =
    "https://en.wikipedia.org/wiki/Cheer_Up_(song)",
44          description = "Lagu ini mengisahkan tentang
    seorang perempuan yang sedang dalam tahap pendekatan
    dengan pria yang sangat posesif, di mana ia meminta
    sedikit ruang agar proses pendekatan menjadi lebih
    menyenangkan dan tidak menekan. Melalui liriknya,
    \"Cheer Up\" menyampaikan pesan untuk tetap semangat dan
    memberikan dukungan kepada orang yang disukai, meskipun
    ada rasa canggung dan ketidakpastian dalam hubungan yang
    sedang berkembang. Lagu ini juga menonjolkan karakter
    ceria dan energik khas TWICE, dengan melodi yang catchy
    dan koreografi yang dinamis."
45      ),
46      Music(
47          id = 6,
48          title = "TT",
49          year = "24 Oktober 2016",
50          imageUrl =
    "https://i.scdn.co/image/ab67616d0000b273387444ab2fc1f08
    dfe7915ab",
51          externalUrl =
    "https://en.wikipedia.org/wiki/TT_(song)",
52          description = "Lagu \"TT\" menggambarkan
    perasaan cinta yang membingungkan dan penuh gejolak.
    Liriknya menceritakan tentang seorang perempuan yang
    merasa kesal dan bingung terhadap perasaan cintanya yang
    semakin menggebu-gebu meskipun ia berusaha menjauh.
    Perasaan tersebut membuatnya merasa seperti \"TT\" -
    sebuah ekspresi emotikon menangis yang juga menjadi
    simbol khas lagu ini. Lagu ini mengungkapkan dilema
    antara ingin mendekat dan sekaligus merasa canggung atau
    takut dalam menghadapi cinta."
53      )
54  )
```

Tabel 10. Source Code Jawaban Soal 1 MusicData.kt

3. Music.kt

```
1   package com.presca.modul4.models
2
3   data class Music(
4       val id: Int,
5       val title: String,
6       val year: String,
7       val imageUrl: String,
```

```
8       val externalUrl: String,
9       val description: String
10  )
```

Tabel 11. Source Code Jawaban Soal 1 Music.kt

4. MusicDetailScreen.kt

```
1   package com.presca.modul4.ui.screens
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.lazy.LazyColumn
5   import androidx.compose.material.icons.Icons
6   import androidx.compose.material.icons.filled.ArrowBack
7   import androidx.compose.material3.*
8   import androidx.compose.runtime.Composable
9   import androidx.compose.ui.Modifier
10  import androidx.compose.ui.draw.clip
11  import androidx.compose.ui.unit.dp
12  import androidx.navigation.NavController
13  import
    com.bumptech.glide.integration.compose.ExperimentalGlid
    eComposeApi
14  import
    com.bumptech.glide.integration.compose.GlideImage
15  import com.presca.modul4.models.Music
16  import
    androidx.compose.foundation.shape.RoundedCornerShape
17
18  @OptIn(ExperimentalGlideComposeApi::class,
    ExperimentalMaterial3Api::class)
19  @Composable
20  fun MusicDetailScreen(music: Music, navController:
    NavController) {
21      Scaffold(
22          topBar = {
23              TopAppBar(
24                  title = { Text(music.title, color =
    MaterialTheme.colorScheme.onPrimary) },
25                  navigationIcon = {
26                      IconButton(onClick = {
    navController.popBackStack() }) {
27                          Icon(Icons.Filled.ArrowBack,
    contentDescription = "Back")
28                      }
29                  },
30                  colors =
```

```
    TopAppBarDefaults.topAppBarColors(containerColor =
    MaterialTheme.colorScheme.surface)
31              )
32          }
33      ) { padding ->
34          LazyColumn(
35              modifier =
    Modifier.padding(padding).padding(16.dp),
36              verticalArrangement =
    Arrangement.spacedBy(12.dp)
37          ) {
38              item {
39                  GlideImage(
40                      model = music.imageUrl,
41                      contentDescription = music.title,
42                      modifier =
    Modifier.fillMaxWidth().height(400.dp).clip(RoundedCorn
    erShape(16.dp))
43                  )
44              }
45              item {
46                  Text("Judul: ${music.title}", style =
    MaterialTheme.typography.titleLarge)
47              }
48              item {
49                  Text("Tanggal Rilis: ${music.year}")
50              }
51              item {
52                  Text("Tentang Lagu Ini:", style =
    MaterialTheme.typography.titleMedium)
53              }
54              item {
55                  Text(music.description, style =
    MaterialTheme.typography.bodyMedium)
56              }
57          }
58      }
59  }
60
```

Tabel 12. Source Code Jawaban Soal 1 MusicDetailScreen.kt

5. MusicListScreen.kt

```
1   package com.presca.modul4.ui.screens
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.lazy.LazyColumn
```

```
5   import androidx.compose.foundation.lazy.items
6   import
    androidx.compose.foundation.shape.RoundedCornerShape
7   import androidx.compose.material3.*
8   import androidx.compose.runtime.Composable
9   import androidx.compose.runtime.collectAsState
10  import androidx.compose.ui.Modifier
11  import androidx.compose.ui.draw.clip
12  import androidx.compose.ui.text.font.FontWeight
13  import androidx.compose.ui.unit.dp
14  import androidx.compose.ui.unit.sp
15  import
    com.bumptech.glide.integration.compose.ExperimentalGlid
    eComposeApi
16  import
    com.bumptech.glide.integration.compose.GlideImage
17  import com.presca.modul4.models.Music
18  import com.presca.modul4.viewmodel.MusicViewModel
19
20  @OptIn(ExperimentalGlideComposeApi::class,
    ExperimentalMaterial3Api::class)
21  @Composable
22  fun MusicListScreen(
23      viewModel: MusicViewModel,
24      onMusicClick: (Music) -> Unit,
25      onExternalClick: (String) -> Unit
26  ) {
27      val musicList =
    viewModel.musicList.collectAsState().value
28
29      Scaffold(
30          topBar = { TopAppBar(title = { Text("List of
    Twice's Best Songs") }) }
31      ) { padding ->
32          LazyColumn(
33              contentPadding = padding,
34              modifier =
    Modifier.fillMaxSize().padding(8.dp)
35          ) {
36              items(musicList) { music ->
37                  Card(
38                      modifier =
    Modifier.fillMaxWidth().padding(vertical = 10.dp),
39                      shape = RoundedCornerShape(16.dp),
40                      elevation =
    CardDefaults.cardElevation(6.dp)
41                  ) {
42                      Row(modifier =
    Modifier.padding(12.dp)) {
```

```
43                          GlideImage(
44                              model = music.imageUrl,
45                              contentDescription =
    music.title,
46                              modifier =
    Modifier.size(120.dp).clip(RoundedCornerShape(12.dp))
47                          )
48                          Spacer(Modifier.width(16.dp))
49                          Column(modifier =
    Modifier.weight(1f)) {
50                              Text(music.title, fontSize
    = 18.sp, style = MaterialTheme.typography.titleMedium)
51                              Text("Tanggal Rilis:
    ${music.year}", fontSize = 14.sp)

52  Spacer(Modifier.height(8.dp))
53                              Text("Tentang Lagu Ini:",
    fontSize = 12.sp, fontWeight = FontWeight.Bold)
54                              Text(music.description,
    fontSize = 12.sp, maxLines = 3)

55  Spacer(Modifier.height(12.dp))
56                              Row(
57                                  modifier =
    Modifier.fillMaxWidth(),
58                                  horizontalArrangement =
    Arrangement.spacedBy(8.dp)
59                              ) {
60                                  Button(onClick = {
    onExternalClick(music.externalUrl) }, modifier =
    Modifier.weight(1f)) {
61                                      Text("Info")
62                                  }
63                                  Button(onClick = {
    onMusicClick(music) }, modifier = Modifier.weight(1f))
    {
64                                      Text("Detail")
65                                  }
66                              }
67                          }
68                      }
69                  }
70              }
71          }
72      }
73  }
```

Tabel 13. Source Code Jawaban Soal 1 MusicListScreen.kt

6. MusicViewModel.kt

```
1   package com.presca.modul4.viewmodel
2
3   import android.util.Log
4   import androidx.lifecycle.ViewModel
5   import com.presca.modul4.data.twiceMusicList
6   import com.presca.modul4.models.Music
7   import kotlinx.coroutines.flow.MutableStateFlow
8   import kotlinx.coroutines.flow.StateFlow
9
10  class MusicViewModel : ViewModel() {
11      private val _musicList =
    MutableStateFlow(twiceMusicList)
12      val musicList: StateFlow<List<Music>> = _musicList
13
14      fun logSelect(music: Music) {
15          Log.d("MusicViewModel", "Musik:
    ${music.title}")
16      }
17
18      fun logDetailClick() {
19          Log.d("MusicViewModel", "Tombol detail
    ditekan")
20      }
21
22      fun logExternalClick(url: String) {
23          Log.d("MusicViewModel", "link website informasi
    musik ditekan: $url")
24      }
25
26      init {
27          Log.d("MusicViewModel", " Musik list berisi
    ${_musicList.value.size} items")
28      }
29  }
30
```

Tabel 14. Source Code Jawaban Soal 1 MusicViewModel.kt

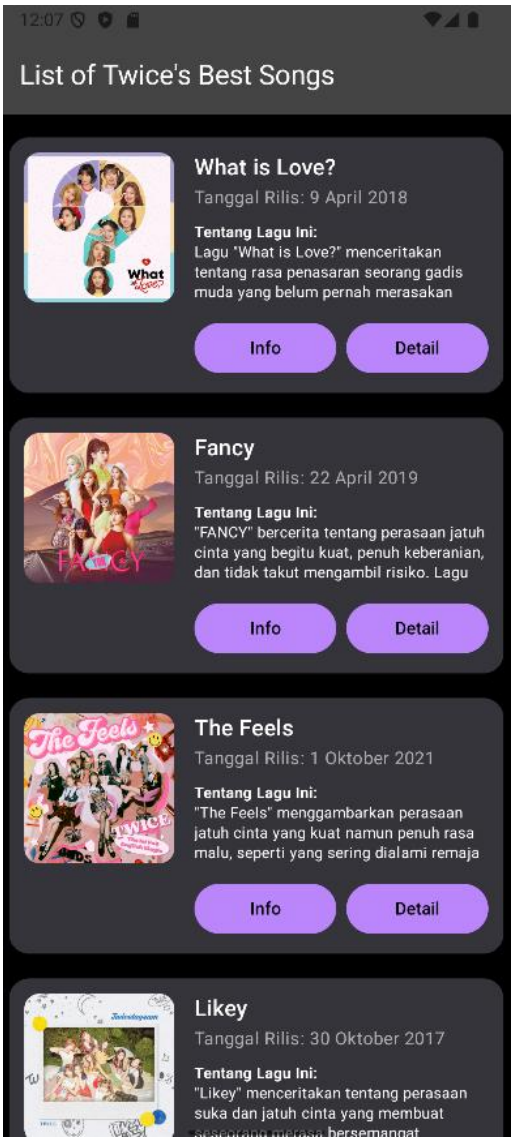
7. MusicViewModelFactory.kt

```
1   package com.presca.modul4.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
```

```
5
6   class MusicViewModelFactory : ViewModelProvider.Factory
    {
7       override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
8           if
    (modelClass.isAssignableFrom(MusicViewModel::class.java
    )) {
9               return MusicViewModel() as T
10          }
11          throw IllegalArgumentException("Unknown
    ViewModel class")
12      }
13  }
```

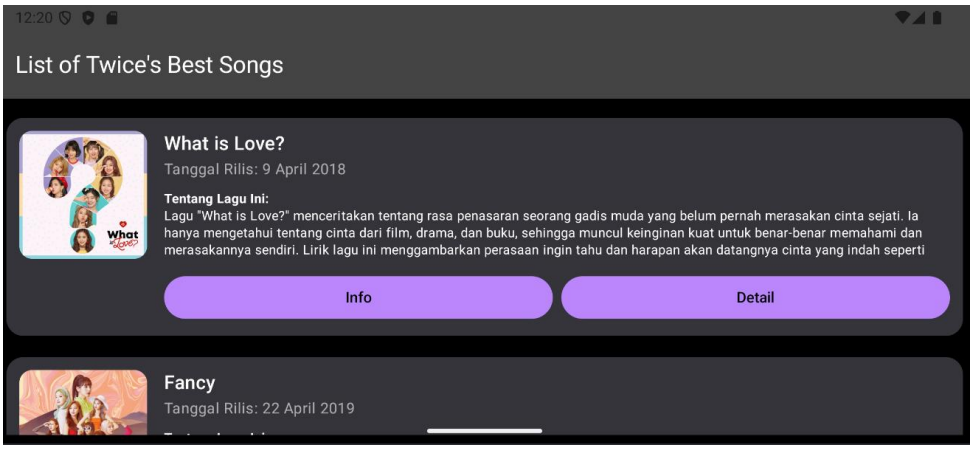
Tabel 15. Source Code Jawaban Soal 1 MusicViewModelFactory.kt

B. Output Program

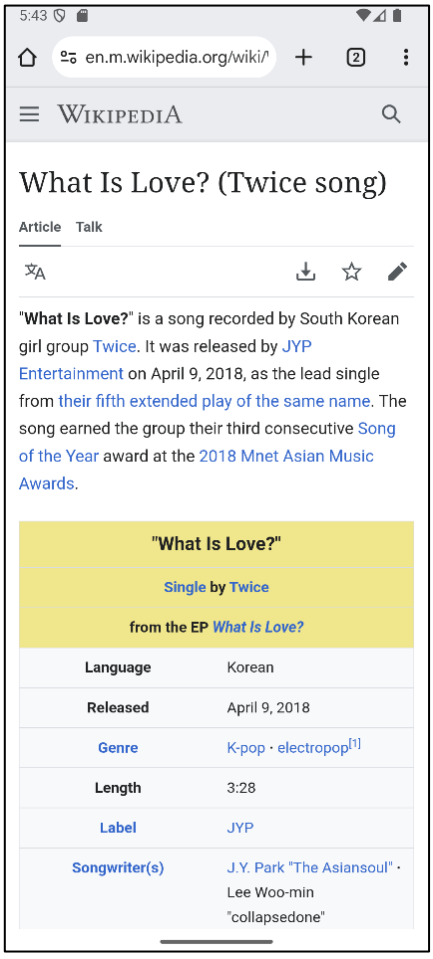


Gambar 21. Screenshot Hasil Jawaban Soal 1 Tampilan List

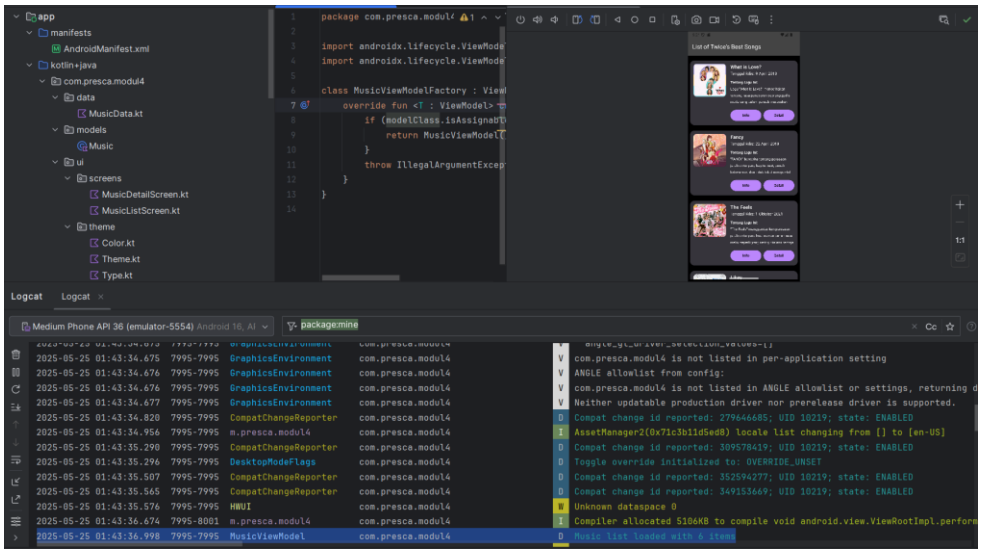
Gambar 22. Screenshot Hasil Jawaban Soal 1 Tampilan Detail



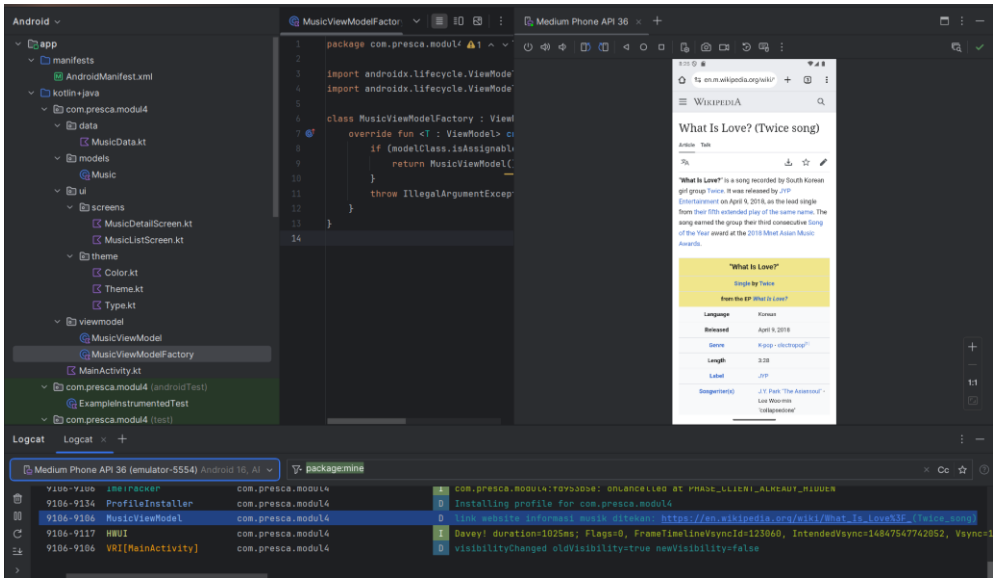
Gambar 23. Screenshot Hasil Jawaban Soal 1 Mode Landscape



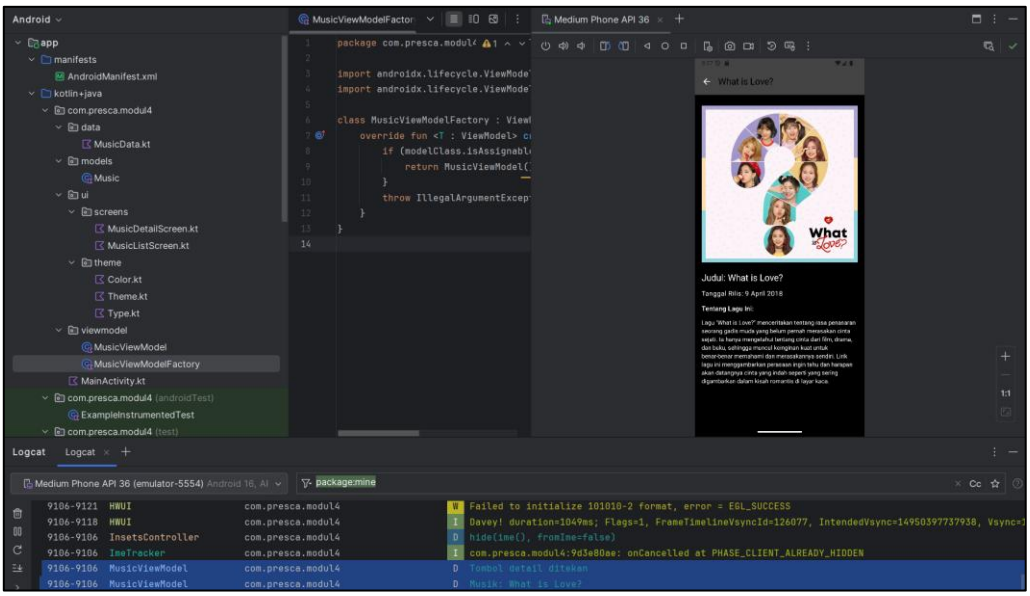
Gambar 24. Screenshot Hasil Jawaban Soal 1 Hasil Tombol Info



Gambar 25. Screenshot Hasil Jawaban Soal 1 Pada Saat Memuat List Musik



Gambar 26. Screenshot Hasil Jawaban Soal 1 Pada Saat Memuat Info Eksternal Musik



Gambar 27. Screenshot Hasil Jawaban Soal 1 Pada Saat Memuat Info Detail Musik

C. Pembahasan

Berikut adalah penjelasan untuk soal nomor 1:

1. MainActivity.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul4`.
- Pada baris [3] hingga [20], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [22], `class MainActivity : ComponentActivity()` ini digunakan sebagai titik awal aplikasi yang akan mengatur tampilan aplikasi.
- Pada baris [23], `private val viewModel: MusicViewModel by viewModels { MusicViewModelFactory() }` ini menginisiasikan `MusicViewModel` menggunakan delegasi `by`

`viewModels` dengan `MusicViewModelFactory`. `ViewModel` digunakan untuk logika bisnis dan data.

- Pada baris [25], `onCreate()` merupakan siklus hidup (lifecycle) dari sebuah activity, yang dimana fungsi ini dipanggil pertama kali saat Activity dibuat.
- Pada baris [26], `super.onCreate(savedInstanceState)` ini memanggil `onCreate()` agar proses inisialisasi standar tetap berjalan.
- Pada baris [27], `enableEdgeToEdge()` digunakan untuk mengaktifkan rendering hingga ke edge layar.
- Pada baris [28], `setContent{…}` digunakan untuk menetapkan tampilan UI dari aplikasi, dan diterapkan tampilan dari `Modul4Theme`.
- Pada baris [29], diterapkan tema khusus aplikasi dari `Modul4Theme`.
- Pada baris [30], `rememberNavController()` digunakan untuk membuat controller navigasi yang mengatur perpindahan antar-screen.
- Pada baris [28], `AppNavigation(navController, viewModel)` berisi logika navigasi aplikasi dan menggunakan viewModel untuk menyimpan logika terkait dengan navigasi.
- Pada baris [38], `AppNavigation` adalah fungsi compose yang digunakan untuk menangani semua screen dan navigasi antar-screen.
- Pada baris [39], `LocalContext.current` ini berisi akses Context Android dari dalam Compose.
- Pada baris [40], `NavHost` merupakan container dari navigasi. `startDestination` ini digunakan untuk mengatur screen awal adalah "music_list".
- Pada baris [41], didefinisikan “music_list” untuk menampilkan daftar musik.
- Pada baris [42], `MusicListScreen(...)` merupakan fungsi `@Composable` yang bertanggung jawab menampilkan daftar musik.

- Pada baris [43], `viewModel = viewModel` digunakan untuk mendapatkan data musik dan menjalankan fungsi logika.
- Pada baris [44], fungsi `onMusicClick` ini akan terpanggil ketika pengguna menekan button “Detail” dan akan menampilkan detail musik.
- Pada baris [45], Dipanggil fungsi `logDetailClick()` dari `viewModel`.
- Pada baris [46], memanggil `logSelect(it)` untuk mencatat lagu apa yang diklik.
- Pada baris [47], `navController.navigate("detail/${it.id}")` ini merupakan navigasi ke halaman dari detail lagu berdasarkan ID.
- Pada baris [49] hingga [51], fungsi `onExternalClick` ini akan terpanggil ketika pengguna menekan button “Info” dan halaman akan berpindah membuka URL di browser.
- Pada baris [55], `composable("detail/{musicId}")` ini digunakan untuk mendefinisikan route untuk tampilan detail musik.
- Pada baris [56], `val musicId = backStackEntry.arguments?.getString("musicId")?.toIntOrNull()` ini akan mengambil nilai parameter `musicId` dari rute yang akan dikirim pada saat navigasi.
- Pada baris [57], `val music = viewModel.musicList.value.find { it.id == musicId }` digunakan untuk mencaari objek di dalam `musicList` dari `viewModel` berdasarkan ID.
- Pada baris [58] dan [59], Jika music ditemukan (tidak null), maka tampilkan `MusicDetailScreen`. Informasi ini ditampilkan sesuai dengan ID.

2. **MusicData.kt:**
    - Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul4.data`
    - Pada baris [3], `import com.presca.modul4.models.Music` digunakan untuk mengimpor kelas `Music` yang memiliki data class yang berisi struktur dari lagu.
    - Pada baris [5], `twiceMusicList` ini digunakan untuk membuat list dari musik yang nantinya akan ditampilkan pada `MusicListScreen`.
    - Pada baris [7] hingga [12], diisi dari list musik pertama dengan format `id` (sesuai urutan lagu), `title` (judul lagu), `year` (tanggal rilis), `imageUrl` (gambar album), `externalUrl` (link yang dapat dibuka untuk informasi lebih jelas di browser, dan `description` (untuk penjelasan lebih detailnya).
    - Pada baris [14] hingga [54], sesuaikan struktur sebelumnya untuk mengisi sisa lagu yang ingin dibuat ke dalam list.

3. **Music.kt:**
    - Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul4.models`
    - Pada baris [3], didefinisikan data dengan nama `Music`.
    - Pada baris [4], didefinisikan juga `id` dalam bentuk integer.
    - Pada baris [5] hingga [9], didefinisikan `title`, `year`, `imageUrl`, `externalUrl`, dan `description` dalam bentuk string.

4. **MusicDetailScreen.kt:**

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul4.ui.screens`
- Pada baris [3] hingga [16], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [18], `@OptIn(…)` ini digunakan untuk mengindikasikan jika kita menggunakan API eksperimental (Glide Compose dan Material3).
- Pada baris [20], `MusicDetailScreen` dibuat dengan parameter `music` untuk data musik, dan `navController` untuk navigasi antar screen.
- Pada baris [21], `Scaffold` digunakan untuk membuat struktur layout utama.
- Pada baris [22], `topBar` digunakan untuk mendefinisikan bar di bagian atas.
- Pada baris [24], `title` nantinya pada bagian ini digunakan untuk menampilkan judul lagu pada bar atas, dan `text` nantinya digunakan untuk menampilkan teks pada judul.
- Pada baris [25], bagian `navigationIcon` ini digunakan untuk mengatur tombol navigasi (seperti tombol kembali).
- Pada baris [26], `onClick` pada bagian ini, `navController.popBackStack()` digunakan untuk kembali ke screen sebelumnya dalam stack navigasi
- Pada baris [38], pada bagian `Icon(Icons.Filled.`*`ArrowBack`*`, contentDescription = "Back")` ini untuk menampilkan ikon panah kembali sebagai ikon dari tombol kembali, dengan deskripsi “Back”.
- Pada baris [30], `colors` pada bagian ini digunakan untuk mengatur warna pada top bar. `containerColor` digunakan untuk mengatur warna latar belakang pada scaffold.

- Pada baris [34], `LazyColumn` merupakan sebuah composable yang digunakan untuk menampilkan daftar item yang dapat digulir secara vertikal.
- Pada baris [35], `modifier` pada bagian ini digunakan untuk mengatur layout dari `LazyColumn`.
- Pada baris [36], `Arrangement.spacedBy(12.dp)` ini digunakan untuk memberi jarak 12dp secara vertikal antar item.
- Pada baris [38], `item` pada bagian ini digunakan untuk mengatur ukuran dan bentuk dari cover musik.
- Pada baris [39], `GlideImage` ini merupakan komponen yang digunakan untuk menampilkan gambar dari URL menggunakan library Glide, serta mengatur ukuran dan bentuk dari gambar tersebut.
- Pada baris [45] hingga [55], `item` pada bagian tersebut mengatur kembali untuk tampilan teks dari judul, tanggal rilis, tentang lagu ini, dan teks deskripsi.

5. MusicListScreen.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul4.ui.screens`
- Pada baris [3] hingga [18], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [20], `@OptIn(…)` ini digunakan untuk mengindikasikan jika kita menggunakan API eksperimental (Glide Compose dan Material3).
- Pada baris [22] hingga [25], `MusicListScreen` dibuat dengan parameter callback `onMusicClick` dan `onExternalClick` untuk interaksi pengguna. `viewModel` digunakan untuk mengambil data lagu.

- Pada baris [27], val musicList = viewModel.musicList.collectAsState().value digunakan untuk mengumpulkan data `musicList` dari `MusicViewModel` menggunakan `State`.
- Pada baris [29], `Scaffold` digunakan untuk membuat struktur layout utama.
- Pada baris [30], `topBar` digunakan untuk mendefinisikan bar di bagian atas. `title` nantinya pada bagian ini digunakan untuk menampilkan judul lagu pada bar atas, dan `text` nantinya digunakan untuk menampilkan teks pada bar atas.
- Pada baris [32], `LazyColumn` merupakan sebuah composable yang digunakan untuk menampilkan daftar item yang dapat digulir secara vertikal.
- Pada baris [34], `modifier` pada bagian ini digunakan untuk mengatur layout dari `LazyColumn`.
- Pada baris [36], `items(musicList)` ini merupakan loop melalui daftar musik untuk membuat item.
- Pada baris [37], `Card()` digunakan sebagai komponen Material Design yang digunakan untuk menampilkan konten terkelompok.
- Pada baris [38], `fillMaxWidth()` ini digunakan agar lebar mengisi parent.
- Pada baris [39], `RoundedCornerShape(16.dp)` ini digunakan untuk mengatur sudut kelengkungan dengan radius 16dp
- Pada baris [40], `cardElevation(6.dp)` digunakan untuk mengatur efek bayangan dengan elevasi 6dp.
- Pada baris [42], `Row` digunakan untuk layout horizontal untuk gambar dan juga teks informasi.
- Pada baris [43], `GlideImage` digunakan untuk menampilkan gambar cover dari URL, serta mengatur ukuran dan bentuk dari gambar tersebut.

- Pada baris [48], `Spacer` digunakan untuk memberi jarak horizontal antara gambar dan juga teks, yang dimana pada baris ini diberi jarak 16dp.
- Pada baris [49], `Column` digunakan untuk menyusun teks dan tombol secara vertikal.
- Pada baris [50], `Text` pada bagian ini akan mengatur tampilan dari judul.
- Pada baris [51] hingga [55], `Text` pada bagian tersebut mengatur kembali untuk tampilan teks dari tanggal rilis, tentang lagu ini, dan teks deskripsi.
- Pada baris [56], `Row` pada bagian ini digunakan untuk menyusun dan mengatur dua tombol (tombol info dan detail) secara horizontal.
- Pada baris [60] hingga [64], bagian ini digunakan untuk mengatur tampilan dari `button` "Info" dan "Detail" serta fungsi dari button tersebut pada saat diklik.

6. MusicViewModel.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `package com.presca.modul4.viewmodel`
- Pada baris [3] hingga [8], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [10], `class MusicViewModel : ViewModel()` ini mendefinisikan `MusicViewModel` yang mewarisi `ViewModel` dan ViewModel digunakan untuk menyimpan dan menglola data lagu yang akan ditampilkan.
- Pada baris [11], `_musicList` ini digunakan untuk menyimpan daftar music dari `twiceMusicList` dengan menggunakan `MutableStateFlow`. `MutableStateFlow()` digunakan untuk membungkus suatu data yang dapat berubah dan diamati (observable).

- Pada baris [12], bagian `musicList` ini akan diekspos ke luar sebagai `StateFlow` atau hanya baca.
- Pada baris [14] dan [15], `logSelect(music: Music)` ini digunakan untuk mencatat ketika pengguna memiliki music tertentu dan menampilkan judul dari music yang diklik atau dipilih.
- Pada baris [18] dan [19], logDetailClick() ini digunakan untuk mencatat jika tombol Detail ditekan dan mendeteksi interaksi pengguna ketika tombol Detail ini ditekan.
- Pada baris [22] dan [23], logExternalClick(url: String) ini digunakan untuk mencatat jika tombol Info ditekan dan mendeteksi interaksi pengguna ketika tombol Info ini ditekan.
- Pada baris [26] dan [27], blok `init` ini digunakan ketika `ViewModel` pertama kali dibuat dan mencatat jumlah item musik yang dimuat.

7. MusicViewModelFactory.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `package com.presca.modul4.viewmodel`
- Pada baris [3] dan [4], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [6], `class MusicViewModelFactory : ViewModelProvider.Factory` digunakan untuk mendefinisikan `MusicViewModelFactory` yang menggunakan intervace `ViewModelProvider.Factory` yang bertujuan untuk membuat `ViewModel` secara custom.
- Pada baris [7], `override fun <T : ViewModel> create(modelClass: Class<T>): T` merupakan implementasi

dari `ViewModelProvider.Factory` yang digunakan untuk membuat dan mengembalikan instance dari `ViewModel`.

- Pada baris [8] dan [9], digunakan untuk mengecek apakah `modelClass` adalah `MusicViewModel`, jika benar maka buat instance baru dari `MusicViewModel`.
- Pada baris [11], `throw IllegalArgumentException("Unknown ViewModel class")` digunakan jika class yang diminta bukan `MusicViewModel`, maka dilempar exception pada baris ini.

Jawaban poin e:

- **Pengertian debugger**

  Debugger merupakan alat bantu yang digunakan untuk membantu menemukan, menganalisis, dan juga memperbaiki kesalahan (bug) pada kode program.

- **Fungsi debugger**
  - Digunakan untuk menemukan bug.
  - Memeriksa dan memahami alur dari eksekusi program.
  - Memeriksa perubahan nilai variabel saat runtime.
  - Menguji bagian kode tertentu, apakah dapat berjalan sesuai ekspetasi.
- **Cara menggunakan debugger**
  1. Memasang Breakpoint dengan cara klik pada bagian sisi kiri nomor baris kode, hingga warna merah muncul di baris yang ingin diperiksa.
  2. Jalankan aplikasi dalam mode Debug.
  3. Aplikasi akan berhenti sementara dan pengguna dapat melakukan debugging.
- **Penjelasan fitur Step Into, Step Over, dan Step Out**
  - Step Into: digunakan untuk masuk ke dalam fungsi atau metode yang sedang dipanggil tadi.

- Step Over: digunakan untuk melanjutkan ke baris berikutnya tanpa masuk ke dalam fungsi yang dipanggil.
- Step Out: Keluar dari fungsi pada saat ini dan kembali ke pemanggilnya.

3) Application Class merupakan kelas dasar Android yang mewakili aplikasi dan lifecycle secara keseluruhan dan dibuat sebelum komponen lain (seperti Activity dan Service). Kelas ini hanya dibuat sekali selama aplikasi berjalan.
   - **Fungsi Application Class**
     - Digunakan untuk menginisialisasi global yang dimana memuat layanan yang harus tersedia di seluruh aplikasi, seperti Room, Firebase, dan lainnya,
     - Penyimpanan state aplikasi, yang dimana menyimpan data yang perlu untuk diakses di seluruh Activity, contohnya seperti ViewModel global.
     - Digunakan untuk memanajemen lifecycle aplikasi, yang dimana Application class memberikan developer kendali penuh terhadap perilaku aplikasi pada berbagai tahap keberadaannya. Berbeda dengan lifecycle komponen individual seperti Activity atau Fragment yang hanya mengelola bagian tertentu dari UI, Application class ini memiliki kemampuan untuk memantau dan merespons perubahan status aplikasi secara global.

SOAL 2

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Pembahasan

Application Class merupakan kelas dasar Android yang mewakili aplikasi dan lifecycle secara keseluruhan dan dibuat sebelum komponen lain (seperti Activity dan Service). Kelas ini hanya dibuat sekali selama aplikasi berjalan.

- **Fungsi Application Class**
  - Digunakan untuk menginisialisasi global yang dimana memuat layanan yang harus tersedia di seluruh aplikasi, seperti Room, Firebase, dan lainnya,
  - Penyimpanan state aplikasi, yang dimana menyimpan data yang perlu untuk diakses di seluruh Activity, contohnya seperti ViewModel global.
  - Digunakan untuk memanajemen lifecycle aplikasi, yang dimana Application class memberikan developer kendali penuh terhadap perilaku aplikasi pada berbagai tahap keberadaannya. Berbeda dengan lifecycle komponen individual seperti Activity atau Fragment yang hanya mengelola bagian tertentu dari UI, Application class ini memiliki kemampuan untuk memantau dan merespons perubahan status aplikasi secara global.

MODUL 5 : Connect to the Internet

SOAL 1

1. 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 bebas, contoh API gratis 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 (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)
    f. Gunakan caching strategy pada Room.
    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.

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

A. Source Code

1. MainActivity.kt

```
1  package com.presca.modul5
2
3  import android.content.Intent
4  import android.net.Uri
5  import android.os.Bundle
```

```
6   import androidx.activity.ComponentActivity
7   import androidx.activity.compose.setContent
8   import androidx.activity.viewModels
9   import androidx.compose.runtime.Composable
10  import androidx.compose.runtime.collectAsState
11  import androidx.compose.runtime.getValue
12  import androidx.compose.ui.platform.LocalContext
13  import androidx.lifecycle.ViewModel
14  import androidx.lifecycle.ViewModelProvider
15  import androidx.navigation.compose.NavHost
16  import androidx.navigation.compose.composable
17  import
    androidx.navigation.compose.rememberNavController
18  import com.presca.modul5.data.local.AppDatabase
19  import com.presca.modul5.data.remote.RetrofitInstance
20  import
    com.presca.modul5.data.repository.CountryRepositoryImpl
21  import
    com.presca.modul5.presentation.screens.CountryDetailScr
    een
22  import
    com.presca.modul5.presentation.screens.CountryListScree
    n
23  import com.presca.modul5.ui.theme.Modul5Theme
24  import
    com.presca.modul5.presentation.theme.ThemeViewModel
25  import
    com.presca.modul5.presentation.viewmodel.CountryViewMod
    el
26  import
    com.presca.modul5.presentation.viewmodel.CountryViewMod
    elFactory
27
28  class MainActivity : ComponentActivity() {
29      private val db by lazy {
30          AppDatabase.getDatabase(this)
31      }
32      private val repository by lazy {
33          CountryRepositoryImpl(RetrofitInstance.api, db)
34      }
35      private val countryViewModel: CountryViewModel by
    viewModels {
36          CountryViewModelFactory(repository)
37      }
38      private val themeViewModel: ThemeViewModel by
    viewModels {
39          object : ViewModelProvider.Factory {
40              override fun <T : ViewModel>
    create(modelClass: Class<T>): T {
```

```
41                 if
   (modelClass.isAssignableFrom(ThemeViewModel::class.java
   )) {
42                     @Suppress("UNCHECKED_CAST")
43                     return
   ThemeViewModel(applicationContext) as T
44                 }
45                 throw IllegalArgumentException("Unknown
   ViewModel class")
46             }
47         }
48     }
49
50     override fun onCreate(savedInstanceState: Bundle?)
   {
51         super.onCreate(savedInstanceState)
52         setContent {
53             val isDarkTheme by
   themeViewModel.isDarkTheme.collectAsState()
54
55             Modul5Theme(darkTheme = isDarkTheme) {
56                 AppNavigation(
57                     viewModel = countryViewModel,
58                     themeViewModel = themeViewModel
59                 )
60             }
61         }
62     }
63 }
64
65 @Composable
66 fun AppNavigation(
67     viewModel: CountryViewModel,
68     themeViewModel: ThemeViewModel
69 ) {
70     val navController = rememberNavController()
71     val context = LocalContext.current
72     val state by viewModel.state.collectAsState()
73     val isDarkTheme by
   themeViewModel.isDarkTheme.collectAsState()
74
75     NavHost(
76         navController = navController,
77         startDestination = "country_list"
78     ) {
79         composable("country_list") {
80             CountryListScreen(
81                 state = state,
82                 onRefresh = {
```

```
    viewModel.refreshCountries() },
83                  onClickDetail = { country ->

84  navController.navigate("detail/${country.id}")
85                  },
86                  onClickInfo = { url ->
87                      try {
88                          context.startActivity(
89                              Intent(Intent.ACTION_VIEW,
    Uri.parse(url))
90                          )
91                      } catch (e: Exception) {
92                          e.printStackTrace()
93                      }
94                  },
95                  onToggleTheme = {
    themeViewModel.toggleTheme() },
96                  isDarkTheme = isDarkTheme
97              )
98          }
99          composable("detail/{id}") { backStackEntry ->
100             val id =
    backStackEntry.arguments?.getString("id")?.toLongOrNull
101 ()
102
103             when (val currentState = state) {
                    is
    CountryViewModel.CountryState.Success -> {
104                     currentState.countries.find { it.id
    == id }?.let { country ->
105                         CountryDetailScreen(
106                             country = country,
107                             navController =
    navController,
108                             onClickInfo = {
109                                 context.startActivity(
110                                     Intent(

111 Intent.ACTION_VIEW,

112 Uri.parse(country.externalUrl)
113                                     )
114                                 )
115                             }
116                         )
117                     } ?: run {
118                         println("Error: Country with ID
    $id not found in current state.")
119                     }
```

```
120                 }
121                 else -> {
122                     println("Error: Cannot display
    detail screen in state: $currentState")
123                 }
124             }
125         }
126     }
127 }
```

Tabel 16. Source Code Jawaban Soal 1 MainActivity.kt

2. data/local/dao/CountryDao.kt

```
1  package com.presca.modul5.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 com.presca.modul5.data.local.entity.CountryEntity
8
9  @Dao
10 interface CountryDao {
11     @Insert(onConflict = OnConflictStrategy.REPLACE)
12     suspend fun insertAll(countries:
   List<CountryEntity>)
13
14     @Query("SELECT * FROM countries")
15     suspend fun getAllCountries(): List<CountryEntity>
16
17     @Query("DELETE FROM countries")
18     suspend fun clearAll()
19
20     @Query("SELECT COUNT(*) FROM countries")
21     suspend fun count(): Int
22 }
```

Tabel 17. Source Code Jawaban Soal 1 CountryDao.kt

3. data/local/entity/CountryEntity.kt

```
1  package com.presca.modul5.data.local.entity
2
3  import androidx.room.Entity
```

```
4   import androidx.room.PrimaryKey
5
6   @Entity(tableName = "countries")
7   data class CountryEntity(
8       @PrimaryKey val id: Long,
9       val name: String,
10      val officialName: String,
11      val flagUrl: String,
12      val region: String,
13      val capital: String,
14      val description: String,
15      val externalUrl: String,
16      val lastUpdated: Long = System.currentTimeMillis()
17  )
```

Tabel 18. Source Code Jawaban Soal 1 CountryEntityc.kt

4. data/local/AppDatabase.kt

```
1   package com.presca.modul5.data.local
2
3   import androidx.room.Database
4   import androidx.room.Room
5   import androidx.room.RoomDatabase
6   import android.content.Context
7   import com.presca.modul5.data.local.dao.CountryDao
8   import
    com.presca.modul5.data.local.entity.CountryEntity
9
10  @Database(
11      entities = [CountryEntity::class],
12      version = 1,
13      exportSchema = false
14  )
15  abstract class AppDatabase : RoomDatabase() {
16      abstract fun countryDao(): CountryDao
17
18      companion object {
19          @Volatile
20          private var INSTANCE: AppDatabase? = null
21
22          fun getDatabase(context: Context): AppDatabase
    {
23              return INSTANCE ?: synchronized(this) {
24                  val instance = Room.databaseBuilder(
25                      context.applicationContext,
26                      AppDatabase::class.java,
```

```
27                          "country_database"
28                  ).build()
29                  INSTANCE = instance
30                  instance
31              }
32          }
33      }
34  }
35
```

Tabel 19. Source Code Jawaban Soal 1 AppDatabase.kt

5. data/mapper/CountryMapper.kt

```
1   package com.presca.modul5.data.mapper
2
3   import
    com.presca.modul5.data.local.entity.CountryEntity
4   import com.presca.modul5.data.remote.response.Country
5   import com.presca.modul5.domain.model.CountryInfo
6   import java.text.DecimalFormat
7
8   object CountryMapper {
9       fun mapResponseToEntity(country: Country, index:
    Long): CountryEntity {
10          return CountryEntity(
11              id = index,
12              name = country.name.common,
13              officialName = country.name.official ?:
    country.name.common,
14              flagUrl = country.flags.png,
15              region = country.region ?: "Unknown",
16              capital = country.capital?.firstOrNull() ?:
    "Unknown",
17              description =
    buildCountryDescription(country),
18              externalUrl =
    generateWikipediaUrl(country.name.common)
19          )
20      }
21
22      fun mapEntityToDomain(entity: CountryEntity):
    CountryInfo {
23          return CountryInfo(
24              id = entity.id,
25              name = entity.name,
26              officialName = entity.officialName,
27              flagUrl = entity.flagUrl,
```

```
28              region = entity.region,
29              capital = entity.capital,
30              description = entity.description,
31              externalUrl = entity.externalUrl
32          )
33      }
34
35      private fun buildCountryDescription(country:
    Country): String {
36          return """
37              |Nama Resmi: ${country.name.official ?:
    country.name.common}
38              |Ibukota: ${country.capital?.firstOrNull()
    ?: "Unknown"}
39              |Region: ${country.region ?: "Unknown"}
40              |Subregion: ${country.subregion ?:
    "Unknown"}
41              |Populasi:
    ${country.population?.formatWithCommas() ?: "Unknown"}
    jiwa
42              |Bahasa Resmi:
    ${country.languages?.values?.joinToString(", ") ?:
    "Unknown"}
43          """.trimMargin()
44      }
45
46      private fun Long.formatWithCommas(): String {
47          return DecimalFormat("#,###").format(this)
48      }
49
50      private fun generateWikipediaUrl(countryName:
    String): String {
51          return
    "https://en.wikipedia.org/wiki/${countryName.replace("
    ", "_")}"
52      }
53  }
54
```

Tabel 20. Source Code Jawaban Soal 1 CountryMapper.kt

6. data/remote/response/Country.kt

```
1   package com.presca.modul5.data.remote.response
2
3   import kotlinx.serialization.SerialName
4   import kotlinx.serialization.Serializable
5
```

```
@Serializable
data class Country(
    val name: Name,
    val flags: Flags,
    val region: String? = null,
    val subregion: String? = null,
    val capital: List<String>? = null,
    val population: Long? = null,
    val languages: Map<String, String>? = null,
    val timezones: List<String>? = null
)

@Serializable
data class Name(
    val common: String,
    val official: String? = null,
    @SerialName("nativeName")
    val nativeNames: Map<String, NativeName>? = null
)

@Serializable
data class NativeName(
    val official: String? = null,
    val common: String? = null
)

@Serializable
data class Flags(
    val png: String,
    val svg: String? = null,
    val alt: String? = null
)
```

Tabel 21. Source Code Jawaban Soal 1 Country.kt

7. data/remote/CountryApiService.kt

```
package com.presca.modul5.data.remote

import com.presca.modul5.data.remote.response.Country
import retrofit2.http.GET

interface CountryApiService {

@GET("v3.1/all?fields=name,flags,region,subregion,capital,population,languages,timezones")
```

```
8       suspend fun getAllCountries(): List<Country>
9   }
```

Tabel 22. Source Code Jawaban Soal 1 CountryApiService.kt

8. data/remote/RetrofitInstance.kt

```
1   package com.presca.modul5.data.remote
2
3   import
    com.jakewharton.retrofit2.converter.kotlinx.serializati
    on.asConverterFactory
4   import kotlinx.serialization.json.Json
5   import okhttp3.MediaType.Companion.toMediaType
6   import okhttp3.OkHttpClient
7   import okhttp3.logging.HttpLoggingInterceptor
8   import retrofit2.Retrofit
9   import java.util.concurrent.TimeUnit
10
11  object RetrofitInstance {
12      private val json = Json {
13          ignoreUnknownKeys = true
14          isLenient = true
15          explicitNulls = false
16      }
17
18      private val loggingInterceptor =
    HttpLoggingInterceptor().apply {
19          level = HttpLoggingInterceptor.Level.BODY
20      }
21
22      private val httpClient = OkHttpClient.Builder()
23          .addInterceptor(loggingInterceptor)
24          .connectTimeout(30, TimeUnit.SECONDS)
25          .readTimeout(30, TimeUnit.SECONDS)
26          .build()
27
28      val api: CountryApiService by lazy {
29          Retrofit.Builder()
30              .baseUrl("https://restcountries.com/")
31              .client(httpClient)

32  .addConverterFactory(json.asConverterFactory("applicati
    on/json".toMediaType()))
33              .build()
34              .create(CountryApiService::class.java)
```

```
35      }
36  }
```

Tabel 23. Source Code Jawaban Soal 1 RetrofitInstance.kt

9. data/repository/CountryRepositoryImpl.kt

```
1   package com.presca.modul5.data.repository
2
3   import com.presca.modul5.data.local.AppDatabase
4   import com.presca.modul5.data.mapper.CountryMapper
5   import com.presca.modul5.data.remote.CountryApiService
6   import com.presca.modul5.domain.model.CountryInfo
7   import
    com.presca.modul5.domain.repository.CountryRepository
8   import kotlinx.coroutines.Dispatchers
9   import kotlinx.coroutines.flow.Flow
10  import kotlinx.coroutines.flow.flow
11  import kotlinx.coroutines.flow.flowOn
12  import kotlinx.coroutines.withContext
13
14  class CountryRepositoryImpl constructor(
15      private val api: CountryApiService,
16      private val db: AppDatabase
17  ) : CountryRepository {
18
19      override fun fetchCountries():
    Flow<List<CountryInfo>> = flow {
20          try {
21              val cachedCountries =
    withContext(Dispatchers.IO) {
22                  db.countryDao().getAllCountries()
23              }
24
25              if (cachedCountries.isNotEmpty()) {
26                  emit(cachedCountries.map {
    CountryMapper.mapEntityToDomain(it) })
27              }
28
29              val countries = api.getAllCountries()
30              val entities = countries.mapIndexed {
    index, country ->

31  CountryMapper.mapResponseToEntity(country,
    index.toLong())
32              }
33
```

```
34              withContext(Dispatchers.IO) {
35                  db.countryDao().clearAll()
36                  db.countryDao().insertAll(entities)
37              }
38
39              emit(entities.map {
    CountryMapper.mapEntityToDomain(it) })
40          } catch (e: Exception) {
41              val cachedCountries =
    withContext(Dispatchers.IO) {
42                  db.countryDao().getAllCountries()
43              }
44              if (cachedCountries.isNotEmpty()) {
45                  emit(cachedCountries.map {
    CountryMapper.mapEntityToDomain(it) })
46              } else {
47                  throw e
48              }
49          }
50      }.flowOn(Dispatchers.IO)
51
52      override suspend fun refreshCountries() {
53          try {
54              val countries = api.getAllCountries()
55              val entities = countries.mapIndexed {
    index, country ->

56  CountryMapper.mapResponseToEntity(country,
    index.toLong())
57              }
58              db.countryDao().clearAll()
59              db.countryDao().insertAll(entities)
60          } catch (e: Exception) {
61              throw e
62          }
63      }
64  }
```

Tabel 24. Source Code Jawaban Soal 1 CountryRepositoryImpl.kt

10. domain/model/CountryInfo.kt

```
1   package com.presca.modul5.domain.model
2
3   data class CountryInfo(
4       val id: Long,
5       val name: String,
```

```
6       val officialName: String,
7       val flagUrl: String,
8       val region: String,
9       val capital: String,
10      val description: String,
11      val externalUrl: String
12  )
```

Tabel 25. Source Code Jawaban Soal 1 CountryInfo.kt

11. domain/repository/CountryRepository.kt

```
1   package com.presca.modul5.domain.repository
2
3   import com.presca.modul5.domain.model.CountryInfo
4   import kotlinx.coroutines.flow.Flow
5
6   interface CountryRepository {
7       fun fetchCountries(): Flow<List<CountryInfo>>
8       suspend fun refreshCountries()
9   }
```

Tabel 26. Source Code Jawaban Soal 1 CountryRepository.kt

12. presentation/screens/CountryDetailScreen.kt

```
1   package com.presca.modul5.presentation.screens
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.rememberScrollState
5   import androidx.compose.foundation.verticalScroll
6   import androidx.compose.material.icons.Icons
7   import androidx.compose.material.icons.filled.ArrowBack
8   import androidx.compose.material.icons.filled.Info
9   import androidx.compose.material3.*
10  import androidx.compose.runtime.Composable
11  import androidx.compose.ui.Modifier
12  import androidx.compose.ui.draw.clip
13  import androidx.compose.ui.unit.dp
14  import androidx.navigation.NavController
15  import
    com.bumptech.glide.integration.compose.ExperimentalGlid
    eComposeApi
16  import
    com.bumptech.glide.integration.compose.GlideImage
```

```
17  import com.presca.modul5.domain.model.CountryInfo
18
19  @OptIn(ExperimentalGlideComposeApi::class,
    ExperimentalMaterial3Api::class)
20  @Composable
21  fun CountryDetailScreen(
22      country: CountryInfo,
23      navController: NavController,
24      onClickInfo: () -> Unit
25  ) {
26      Scaffold(
27          topBar = {
28              TopAppBar(
29                  title = { Text(country.name) },
30                  navigationIcon = {
31                      IconButton(onClick = {
    navController.popBackStack() }) {
32                          Icon(Icons.Default.ArrowBack,
    contentDescription = "Kembali")
33                      }
34                  },
35                  actions = {
36                      IconButton(onClick = onClickInfo) {
37                          Icon(Icons.Default.Info,
    contentDescription = "Info")
38                      }
39                  }
40              )
41          }
42      ) { padding ->
43          Column(
44              modifier = Modifier
45                  .padding(padding)
46                  .padding(16.dp)
47                  .verticalScroll(rememberScrollState())
48          ) {
49              GlideImage(
50                  model = country.flagUrl,
51                  contentDescription = "Bendera
    ${country.name}",
52                  modifier = Modifier
53                      .fillMaxWidth()
54                      .height(200.dp)
55                      .clip(MaterialTheme.shapes.medium)
56              )
57
58              Spacer(modifier = Modifier.height(24.dp))
59
60              Text(
```

```
61                  text = "Nama Resmi:",
62                  style =
    MaterialTheme.typography.labelLarge
63              )
64              Text(
65                  text = country.officialName,
66                  style =
    MaterialTheme.typography.bodyLarge,
67                  modifier = Modifier.padding(bottom =
    16.dp)
68              )
69
70              Text(
71                  text = "Informasi:",
72                  style =
    MaterialTheme.typography.labelLarge
73              )
74              Text(
75                  text = country.description,
76                  style =
    MaterialTheme.typography.bodyLarge,
77                  modifier = Modifier.padding(bottom =
    16.dp)
78              )
79
80              Button(
81                  onClick = onClickInfo,
82                  modifier = Modifier.fillMaxWidth()
83              ) {
84                  Text("Buka Info Lengkap")
85              }
86          }
87      }
88  }
```

Tabel 27. Source Code Jawaban Soal 1 CountryDetailScreen.kt

13. presentation/screens/CountryListScreen.kt

```
1   package com.presca.modul5.presentation.screens
2
3   import androidx.compose.foundation.layout.*
4   import androidx.compose.foundation.lazy.LazyColumn
5   import androidx.compose.foundation.lazy.items
6   import
    androidx.compose.foundation.shape.RoundedCornerShape
7   import androidx.compose.material.icons.Icons
```

```
8  import androidx.compose.material.icons.filled.*
9  import androidx.compose.material3.*
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Alignment
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.draw.clip
14 import androidx.compose.ui.unit.dp
15 import androidx.compose.ui.unit.sp
16 import
   com.bumptech.glide.integration.compose.ExperimentalGlid
   eComposeApi
17 import
   com.bumptech.glide.integration.compose.GlideImage
18 import com.presca.modul5.domain.model.CountryInfo
19 import
   com.presca.modul5.presentation.viewmodel.CountryViewMod
   el
20
21 @OptIn(ExperimentalGlideComposeApi::class,
   ExperimentalMaterial3Api::class)
22 @Composable
23 fun CountryListScreen(
24     state: CountryViewModel.CountryState,
25     onRefresh: () -> Unit,
26     onClickDetail: (CountryInfo) -> Unit,
27     onClickInfo: (String) -> Unit,
28     onToggleTheme: () -> Unit,
29     isDarkTheme: Boolean
30 ) {
31     Scaffold(
32         topBar = {
33             CenterAlignedTopAppBar(
34                 title = { Text("Negara di Dunia") },
35                 navigationIcon = {
36                     IconButton(onClick = { /* TODO */
   }) {
37                         Icon(Icons.Filled.Public,
   contentDescription = null)
38                     }
39                 },
40                 actions = {
41                     IconButton(onClick = onToggleTheme)
   {
42                         Icon(
43                             imageVector = if
   (isDarkTheme) Icons.Filled.LightMode else
   Icons.Filled.DarkMode,
44                             contentDescription =
   "Toggle Theme"
```

```
45                          )
46                      }
47                  }
48              )
49          }
50      ) { innerPadding ->
51          Box(modifier = Modifier.padding(innerPadding))
    {
52              when (state) {
53                  is
    CountryViewModel.CountryState.Loading -> LoadingView()
54                  is CountryViewModel.CountryState.Error
    -> ErrorView(state.message, onRefresh)
55                  is
    CountryViewModel.CountryState.Success ->
56                      CountryListView(
57                          countries = state.countries,
58                          onClickDetail = onClickDetail,
59                          onClickInfo = onClickInfo
60                      )
61              }
62          }
63      }
64  }
65
66  @Composable
67  fun LoadingView() {
68      Box(
69          modifier = Modifier.fillMaxSize(),
70          contentAlignment = Alignment.Center
71      ) {
72          CircularProgressIndicator()
73      }
74  }
75
76  @Composable
77  fun ErrorView(message: String, onRefresh: () -> Unit) {
78      Column(
79          modifier = Modifier.fillMaxSize(),
80          verticalArrangement = Arrangement.Center,
81          horizontalAlignment =
    Alignment.CenterHorizontally
82      ) {
83          Text(message)
84          Spacer(modifier = Modifier.height(16.dp))
85          Button(onClick = onRefresh) {
86              Text("Coba Lagi")
87          }
88      }
```

```
89  }
90
91  @OptIn(ExperimentalGlideComposeApi::class)
92  @Composable
93  fun CountryListView(
94      countries: List<CountryInfo>,
95      onClickDetail: (CountryInfo) -> Unit,
96      onClickInfo: (String) -> Unit
97  ) {
98      LazyColumn(
99          modifier = Modifier.fillMaxSize(),
100         contentPadding = PaddingValues(8.dp)
101     ) {
102         items(countries) { country ->
103             CountryCard(
104                 country = country,
105                 onClickDetail = {
    onClickDetail(country) },
106                 onClickInfo = {
    onClickInfo(country.externalUrl) }
107             )
108         }
109     }
110 }
111
112 @OptIn(ExperimentalGlideComposeApi::class)
113 @Composable
114 fun CountryCard(
115     country: CountryInfo,
116     onClickDetail: () -> Unit,
117     onClickInfo: () -> Unit
118 ) {
119     Card(
120         modifier = Modifier
121             .fillMaxWidth()
122             .padding(vertical = 8.dp),
123         shape = RoundedCornerShape(12.dp)
124     ) {
125         Column(modifier = Modifier.padding(12.dp)) {
126             Row(verticalAlignment =
    Alignment.CenterVertically) {
127                 GlideImage(
128                     model = country.flagUrl,
129                     contentDescription = country.name,
130                     modifier = Modifier
131                         .size(80.dp)
132                         .clip(RoundedCornerShape(8.dp))
133                 )
134                 Spacer(modifier =
```

```
    Modifier.width(16.dp))
135                 Column(modifier = Modifier.weight(1f))
    {
136                     Text(country.name, fontSize =
    18.sp)
137                     Text(
138                         text = "Lokasi:
    ${country.region}",
139                         fontSize = 14.sp
140                     )
141                 }
142             }
143             Spacer(modifier = Modifier.height(12.dp))
144             Row(
145                 modifier = Modifier.fillMaxWidth(),
146                 horizontalArrangement =
    Arrangement.spacedBy(8.dp)
147             ) {
148                 Button(
149                     onClick = onClickInfo,
150                     modifier = Modifier.weight(1f)
151                 ) {
152                     Text("Info")
153                 }
154                 Button(
155                     onClick = onClickDetail,
156                     modifier = Modifier.weight(1f)
157                 ) {
158                     Text("Detail")
159                 }
160             }
161         }
162     }
163 }
```

Tabel 28. Source Code Jawaban Soal 1 CountryListScreen.kt

14. presentation/theme/Theme.kt

```
1   package com.presca.modul5.ui.theme
2
3   import androidx.compose.foundation.isSystemInDarkTheme
4   import androidx.compose.material3.MaterialTheme
5   import androidx.compose.material3.darkColorScheme
6   import androidx.compose.material3.lightColorScheme
7   import androidx.compose.runtime.Composable
8
```

```
9   private val DarkColorScheme = darkColorScheme(
10      primary = LightNavyBlue,
11      secondary = PurpleGrey80,
12      tertiary = Pink80
13  )
14
15  private val LightColorScheme = lightColorScheme(
16      primary = NavyBlue,
17      secondary = PurpleGrey40,
18      tertiary = Pink40
19  )
20
21  @Composable
22  fun Modul5Theme(
23      darkTheme: Boolean = isSystemInDarkTheme(),
24      content: @Composable () -> Unit
25  ) {
26      val colorScheme = if (darkTheme) DarkColorScheme
    else LightColorScheme
27
28      MaterialTheme(
29          colorScheme = colorScheme,
30          typography = Typography,
31          content = content
32      )
33  }
34
```

Tabel 29. Source Code Jawaban Soal 1 Theme.kt

15. presentation/theme/ThemeViewModel.kt

```
1   package com.presca.modul5.presentation.theme
2
3   import android.content.Context
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.viewModelScope
6   import androidx.datastore.core.DataStore
7   import androidx.datastore.preferences.core.Preferences
8   import
    androidx.datastore.preferences.core.booleanPreferencesK
    ey
9   import androidx.datastore.preferences.core.edit
10  import
    androidx.datastore.preferences.preferencesDataStore
11  import kotlinx.coroutines.flow.MutableStateFlow
12  import kotlinx.coroutines.flow.StateFlow
```

```
13  import kotlinx.coroutines.flow.asStateFlow
14  import kotlinx.coroutines.flow.map
15  import kotlinx.coroutines.launch
16
17  private val Context.dataStore: DataStore<Preferences>
    by preferencesDataStore(name = "theme_preferences")
18
19  class ThemeViewModel(private val applicationContext:
    Context) : ViewModel() {
20
21      private val IS_DARK_THEME_KEY =
    booleanPreferencesKey("is_dark_theme")
22
23      private val _isDarkTheme = MutableStateFlow(false)
24      val isDarkTheme: StateFlow<Boolean> =
    _isDarkTheme.asStateFlow()
25
26      init {
27          viewModelScope.launch {
28              applicationContext.dataStore.data
29                  .map { preferences ->
30                      preferences[IS_DARK_THEME_KEY] ?:
    false
31                  }
32                  .collect { isDark ->
33                      _isDarkTheme.value = isDark
34                  }
35          }
36      }
37
38      fun toggleTheme() {
39          viewModelScope.launch {
40              val currentTheme = _isDarkTheme.value
41              applicationContext.dataStore.edit {
    preferences ->
42                  preferences[IS_DARK_THEME_KEY] =
    !currentTheme
43              }
44          }
45      }
46  }
47
```

Tabel 30. Source Code Jawaban Soal 1 ThemeViewModel.kt

16. presentation/theme/CountryViewModel.kt

```
1   package com.presca.modul5.presentation.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.viewModelScope
5   import com.presca.modul5.domain.model.CountryInfo
6   import
    com.presca.modul5.domain.repository.CountryRepository
7   import kotlinx.coroutines.flow.MutableStateFlow
8   import kotlinx.coroutines.flow.StateFlow
9   import kotlinx.coroutines.flow.asStateFlow
10  import kotlinx.coroutines.launch
11
12  class CountryViewModel(private val repository:
    CountryRepository) : ViewModel() {
13      sealed class CountryState {
14          object Loading : CountryState()
15          data class Success(val countries:
    List<CountryInfo>) : CountryState()
16          data class Error(val message: String) :
    CountryState()
17      }
18
19      private val _state =
    MutableStateFlow<CountryState>(CountryState.Loading)
20      val state: StateFlow<CountryState> =
    _state.asStateFlow()
21
22      init {
23          fetchCountries()
24      }
25
26      fun fetchCountries() {
27          viewModelScope.launch {
28              _state.value = CountryState.Loading
29              try {
30                  repository.fetchCountries().collect {
    countries ->
31                      _state.value = if
    (countries.isEmpty()) {
32                          CountryState.Error("Tidak ada
    data negara yang ditemukan")
33                      } else {
34                          CountryState.Success(countries)
35                      }
36                  }
37              } catch (e: Exception) {
38                  _state.value = CountryState.Error(
39                      "Gagal memuat data: ${e.message ?:
    "Terjadi kesalahan"}"
```

```
                )
            }
        }
    }

    fun refreshCountries() {
        viewModelScope.launch {
            _state.value = CountryState.Loading
            try {
                repository.refreshCountries()
                fetchCountries()
            } catch (e: Exception) {
                _state.value = CountryState.Error(
                    "Gagal menyegarkan data:
${e.message ?: "Terjadi kesalahan"}"
                )
            }
        }
    }
}
```

Tabel 31. Source Code Jawaban Soal 1 CountryViewModel.kt

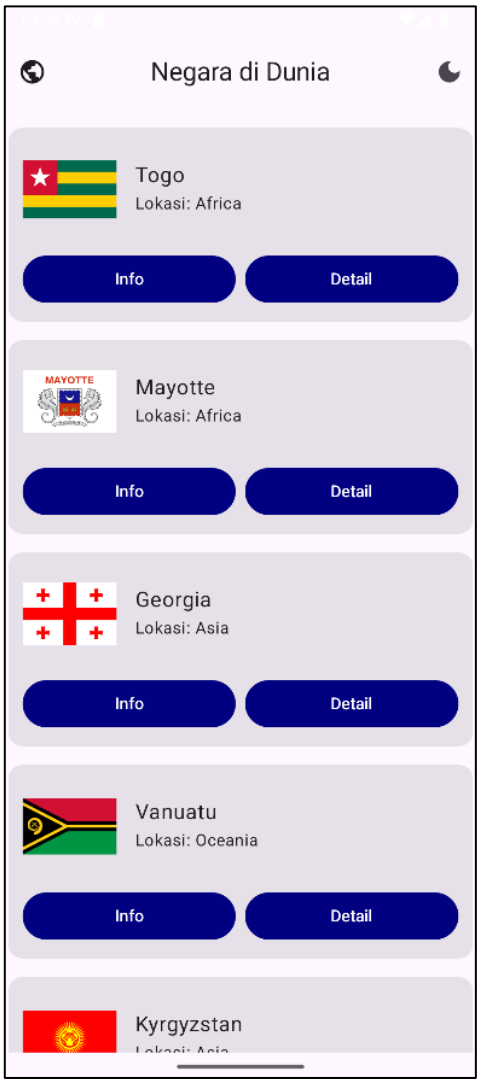
17. presentation/theme/CountryViewModelFactory.kt

```
package com.presca.modul5.presentation.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import
com.presca.modul5.domain.repository.CountryRepository

class CountryViewModelFactory(
    private val repository: CountryRepository
) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass:
Class<T>): T {
        if
(modelClass.isAssignableFrom(CountryViewModel::class.jav
a)) {
            @Suppress("UNCHECKED_CAST")
            return CountryViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown
ViewModel class")
    }
}
```

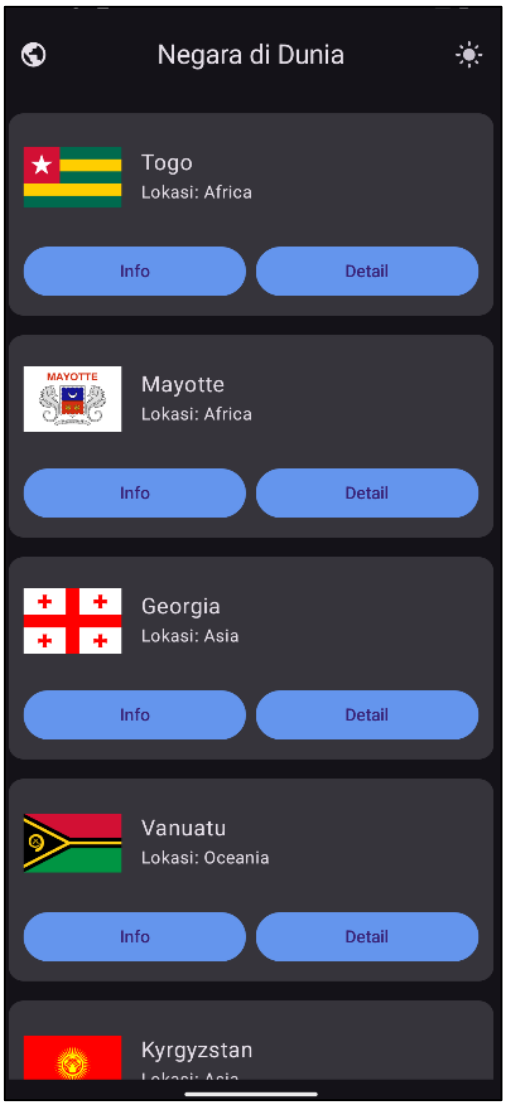
18	

Tabel 32. Source Code Jawaban Soal 1 CountryViewModelFactory.kt

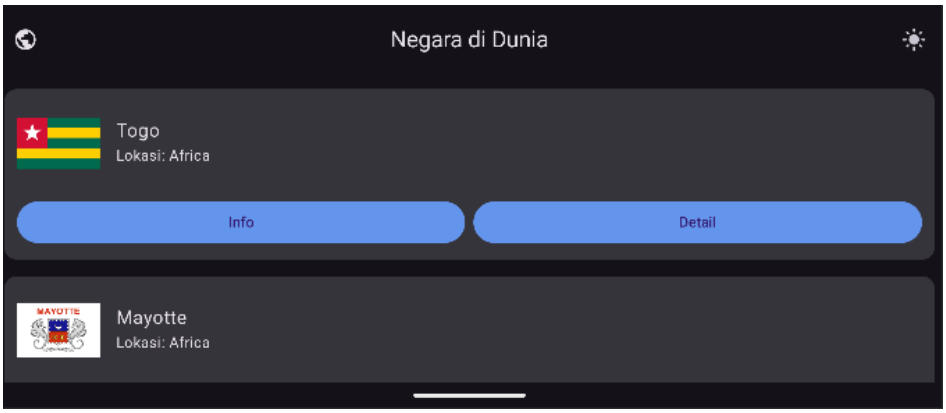
B. Output Program



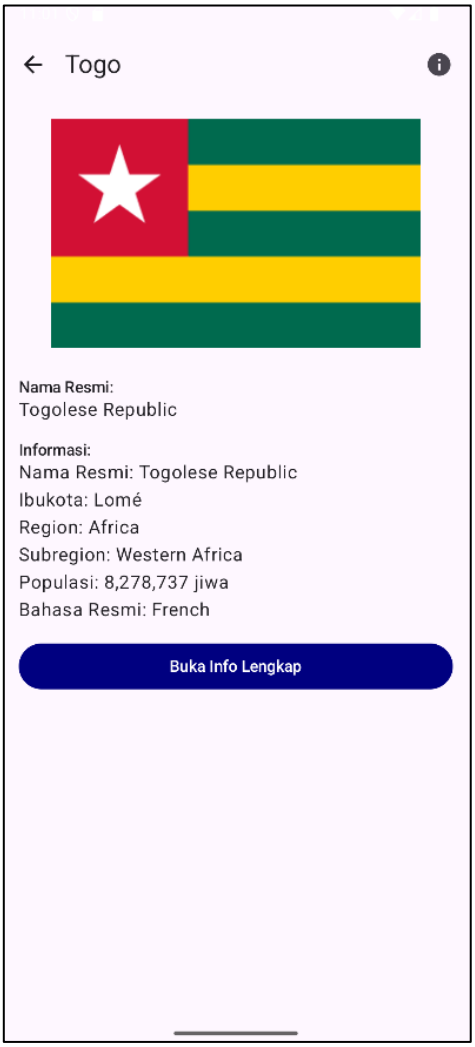
Gambar 28. Screenshot Hasil Jawaban Soal 1 Tampilan List



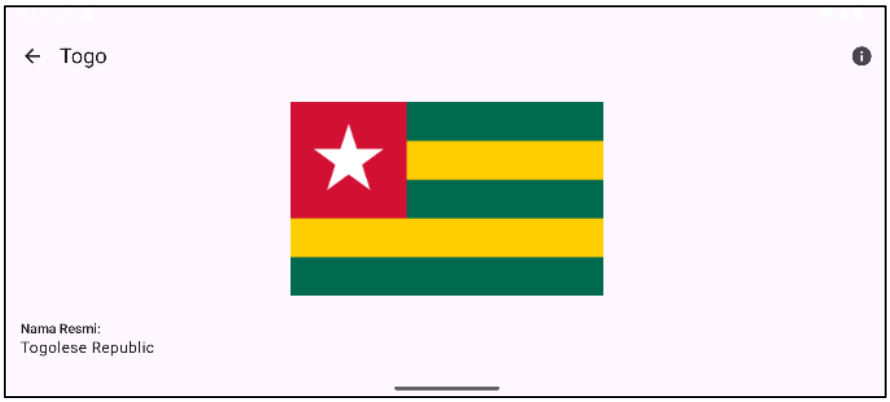
Gambar 29. Screenshot Hasil Jawaban Soal 1 Tampilan List Dark Mode



Gambar 30. Screenshot Hasil Jawaban Soal 1 Tampilan List Landscape



Gambar 31. Screenshot Hasil Jawaban Soal 1 Tampilan Detail



Gambar 32. Screenshot Hasil Jawaban Soal 1 Tampilan Detail Landscape

Gambar 33. Screenshot Hasil Jawaban Soal 1 Tampilan Detail Dark Mode

Gambar 34. Screenshot Hasil Jawaban Soal 1 Hasil Tombol Info

C. Pembahasan

Berikut adalah penjelasan untuk soal nomor 1:

1. **MainActivity.kt:**
    - Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5`

- Pada baris [3] hingga [26], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [28], `class MainActivity : ComponentActivity()` ini digunakan sebagai titik awal aplikasi yang akan mengatur tampilan aplikasi.
- Pada baris [29], `private val db by lazy { AppDatabase.getDatabase(this)` digunakan untuk mendeklarasikan properti db yang akan menampung instance database Room (`AppDatabase`)
- Pada baris [32], `private val repository by lazy { CountryRepositoryImpl(RetrofitInstance.api, db)` digunakan untuk mendeklarasikan properti repository yang menampung instance dari `CountryRepositoryImpl`.
- Pada baris [35], `private val themeViewModel: ThemeViewModel by viewModels` digunakan untuk mendeklarasikan properti `themeViewModel.`
- Pada baris [65] dan [66], `@Composable` dan `fun AppNavigation` pada bagian ini merupakan fungsi Composable yang bertanggung jawab untuk mendefinisikan grafik navigasi pada aplikasi.
- Pada baris [71], `val context = LocalContext.current` digunakan untuk mendapatkan `Context` saat ini dalam lingkungan Composable.
- Pada baris [72], `val state by viewModel.state.collectAsState()` digunakan untuk mengamati state data negara dari `CountryViewModel`.
- Pada baris [75], `NavHost` adalah komponen inti dari navigasi Compose yang bertanggung jawab untuk menampilkan Composable yang sesuai dengan rute navigasi saat ini.

- Pada baris [79], `composable("country_list")` pada bagian digunakan untuk mendefinisikan sebuah "tujuan" navigasi (destination) untuk layar daftar negara.
- Pada baris [99], `composable("detail/{id}") { backStackEntry -> ... }` pada bagian ini digunakan untuk mendefinisikan tujuan navigasi untuk layar detail negara.

2. CountryDao.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local.dao`
- Pada baris [3] hingga [7], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [9], `@Dao` adalah menandakan bahwa CountryDao adalah antarmuka Dao Room yang dimana Room akan memproses antarmuka ini dan menggenerasikan kode yang diperlukan untuk interaksi dengan database.
- Pada baris [11], `@Insert` adalah menandakan bahwa nantinya pada bagian ini akan menyisipkan data pada bagian ini.
- Pada baris [14], `@Query` merupakan argument berupa string SQL, yang dimana pada bagian ini untuk mengambil semua kolom bernama `countries`.
- Pada baris [17], Query digunakan untuk mengapus semua baris tabel `countries`.
- Pada baris [20], Query digunakan untuk menghitung semua baris tabel `countries`.

3. **CountryEntity.kt:**
    - Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local.entity`
    - Pada baris [3] dan [4], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
    - Pada baris [6], `@Entity` digunakan untuk memberitahu Room bahwa kelas ini adalah sebuah entitas database (yaitu, akan dipetakan ke sebuah tabel).
    - Pada baris [7], `data class CountryEntity` digunakan untuk mendeklarasikan sebuah data class Kotlin.
    - Pada baris [8], `@PrimaryKey` pada bagian ini menandai bahwa properti id sebagai Primary Key untuk tabel `countries`.
    - Pada baris [9] hingga [16], berbagai property ditetapkan di dalam tabel `countries`.

4. **AppDatabase.kt:**
    - Pada intinya, `AppDatabase` ini akan mengonfigurasi database, mengaitkan entitas dengan tabel, menyediakan akses ke DAO, dan memastikan bahwa hanya ada satu instance database yang berjalan di seluruh aplikasi, yang merupakan praktik terbaik untuk efisiensi dan mencegah masalah database.
    - Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local`
    - Pada baris [3] hingga [8], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.

- Pada baris [10], `@Database` digunakan untuk konfigurasi utama untuk database Room.
- Pada baris [15], `abstract class AppDatabase : RoomDatabase()` digunakan untuk mendefinisikan kelas `AppDatabase` sebagai kelas `abstract`.
- Pada baris [18], `companion object` mendefinisikan bahwa semua anggota yang dideklarasikan di dalam `companion object` dapat diakses langsung menggunakan nama kelas, tanpa perlu membuat instance kelas tersebut.
- Pada baris [22], `fun getDatabase(context: Context): AppDatabase {…}` ini dgunakan untuk mendapatkan satu-satunya instance dari `AppDatabase`.

5. CountryMapper.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam package `com.presca.modul5.data.local`
- Pada baris [3] hingga [6], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [8], `object CountryMapper {…}` ini digunakan untuk Deklarasi object dalam Kotlin membuat sebuah singleton, yang dimana hanya akan ada satu instance dari `CountryMapper` di seluruh aplikasi.
- Pada baris [9], `fun mapResponseToEntity(country: Country, index: Long): CountryEntity { ..}` digunakan untuk mengkonversi objek `Country` (dari API) menjadi objek `CountryEntity` (untuk database Room).

- Pada baris [22], `fun mapEntityToDomain(entity: CountryEntity): CountryInfo {...}` digunakan untuk mengkonversi objek `CountryEntity` (dari database) menjadi objek `CountryInfo` (untuk lapisan domain/UI).
- Pada baris [35], `private fun buildCountryDescription(country: Country): String {...}` digunakan untuk membuat string deskripsi yang diformat dari objek `Country` yang lebih lengkap.
- Pada baris [46], `private fun Long.formatWithCommas(): String {…}` pada bagian ini digunakan untuk memformat angka Long (seperti populasi) dengan pemisah ribuan.
- Pada baris [50], `private fun generateWikipediaUrl(countryName: String): String {…}` digunakan untuk membuat URL Wikipedia berdasarkan nama negara.

6. Country.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.data.remote.response`.
- Pada baris [3] dan [4], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [6], `@Serializable` digunakan untuk memberitahu KotlinX Serialization bahwa `Country` adalah kelas yang dapat diserialisasi. Room akan menggenerasikan kode untuk mengkonversi objek `Country` ke/dari representasi JSON.
- Pada baris [7], `data class Country(...)` merepresentasikan struktur data utama untuk satu negara yang diterima dari API.

- Pada baris [19], `data class Name(…)` digunakan untuk merepresentasikan berbagai format nama negara.
- Pada baris [27], `data class NativeName(...)` digunakan untuk merepresentasikan nama asli dalam format resmi dan umum.
- Pada baris [33], `data class Flags(...)` Merepresentasikan berbagai format URL bendera dan teks alternatif.

7. CountryApiService.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.data.remote`.
- Pada baris [3] dan [4], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [6], `interface CountryApiService` digunakan untuk mendefinisikan antarmuka `CountryApiService`.
- Pada baris [7], `@GET("v3.1/all?fields=name,flags,region,subregion,capital,population,languages,timezones")` digunakan untuk mengirim permintaan HTTP GET ke server sesuai dengan yang diminta (misal bagian ini akan meminta nama, flags, region, hingga timezones).
- Pada baris [8], `suspend fun getAllCountries(): List<Country>` merupakan metode yang akan melakukan panggilan API.

8. RetrofitInstance.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.data.remote`.
- Pada baris [3] hingga [9], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [11], `object RetrofitInstance {...}` mendefinisikan bahwa hanya ada satu instance dari RetrofitInstance yang akan dibuat di seluruh aplikasi.
- Pada baris [12], `private val json = Json { ... }` merupakan dari bagian konfigurasi untuk bagaimana JSON akan diurai (parse).
- Pada baris [18], `private val loggingInterceptor = HttpLoggingInterceptor().apply { level = HttpLoggingInterceptor.Level.BODY }` digunakan untuk mengatur tingkat logging dan mencatat seluruh body permintaan dan respons.
- Pada baris [22], `private val httpClient = OkHttpClient.Builder().addInterceptor(loggingInterceptor) .connectTimeout(30, TimeUnit.SECONDS) .readTimeout(30, TimeUnit.SECONDS) .build()` digunakan untuk membangun instance OkHttpClient yang akan digunakan oleh Retrofit.
- Pada baris [28], `val api: CountryApiService by lazy {...}` digunakan untuk mendefinisikan properti API (termasuk endpoint) yang akan menyediakan instance `CountryApiService` yang siap pakai.

9. CountryrepositoryImpt.kt:

- Pada intinya, CountryRepositoryImpl.kt adalah implementasi konkret dari antarmuka CountryRepository (yang ada di lapisan domain) dan berfungsi

untuk mengabstraksi asal-usul data (apakah dari jaringan, database lokal, atau memori), menerapkan logika bisnis seperti strategi caching, dan menyediakan data yang konsisten ke lapisan di atasnya (ViewModel).

10. CountryInfo.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.domain.model`
- Pada baris [3], `data class CountryInfo(...)` digunakan untuk mendefinisikan struktur data sebuah objek `CountryInfo` dan secara otomatis menyediakan fungsi-fungsi dasar yang sangat sering dibutuhkan saat bekerja dengan data.

11. CountryRepository.kt:

- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.domain.repository`
- Pada baris [6], `interface CountryRepository {}` digunakan untuk mendefinisikan metode yang fungsionalitas yang berkaitan dengan data negara, tanpa menyediakan implementasi (detail bagaimana fungsionalitas itu bekerja).

12. CountryDetailScreen.kt:

- Pada intinya, `CountryDetailScreen.kt` memiliki berbagai fungsi `@Composable` yang mendefinisikan tampilan antarmuka pengguna (UI) untuk layar detail negara. Bagian ini menampilkan informasi rinci tentang

sebuah negara, seperti bendera, nama resmi, dan deskripsi pada tiap-tiap negara.

13. CountryListScreen.kt:

- Pada intinya, `CountryListScreen.kt` memiliki berbagai fungsi `@Composable` yang mendefinisikan tampilan antarmuka pengguna (UI) untuk layar daftar (list) negara. Bagian ini bertanggung jawab untuk menampilkan daftar negara, menunjukkan status loading atau error, dan memungkinkan interaksi seperti menyegarkan data atau beralih tema.

14. CountryViewModel.kt:

- File `CountryViewModel.kt` ini berguna untuk mengelola state UI terkait data negara, berinteraksi dengan `CountryRepository` untuk mengambil data, dan menyediakan state ini ke UI melalui `StateFlow` agar UI dapat bereaksi secara reaktif terhadap perubahan data, loading, atau error.

15. CountryViewModelFactory.kt:

- File `CountryViewModelFactory.kt` ini berfungsi sebagai pabrik (factory) untuk membuat instance dari `CountryViewModel.` `ViewModelFactory` menjadi sangat penting ketika `ViewModel` memiliki dependensi di konstruktornya (misalnya, `CountryRepository`)
- Pada baris [1], dideklarasikan nama package file Kotlin yang dikelompokkan file ini ke dalam `package com.presca.modul5.presentation.viewmodel`

- Pada baris [3] hingga [5], `import` adalah perintah yang digunakan untuk mengimpor kelas, fungsi, atau objek dari package lain tanpa harus menyebutkan path lengkapnya.
- Pada baris [8], `(private val repository: CountryRepository)` digunakan sebagai konstruktor utama untuk `CountryViewModelFactory` yang digunakan untuk menerima sebuah instance dari `CountryRepository`.
- Pada baris [10], create dalam `ViewModelProvider.Factory` berfungsi sebagai jantung dari proses inisialisasi `ViewModel` yang memiliki dependensi kustom, ketika sistem Android perlu membuat instance `ViewModel` baru (misalnya, saat Activity pertama kali dibuat atau setelah rotasi layar), ia akan memanggil metode create ini, meneruskan objek Class dari `ViewModel` yang diinginkan (misalnya, `CountryViewModel::class.java`) sebagai parameter `modelClass`.

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

https://github.com/Prescaa/Kuliah/tree/master/Praktikum%20Mobile