LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Muhammad Ibnu Sina **NIM. 2310817210009**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Muhammad Ibnu Sina
NIM : 2310817210009

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar	Muti`a Maulida S.Kom M.T.I
NIM. 2210817210026	NIP. 19881027 201903 20 13

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

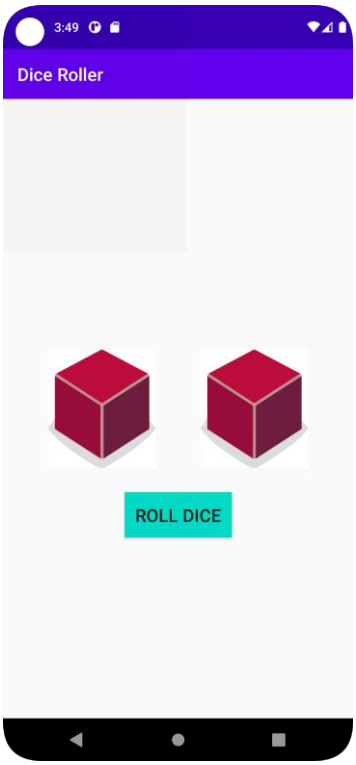
MODUL 1 : ANDROID BASIC WITH KOTLIN

SOAL 1

Soal Praktikum:

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:

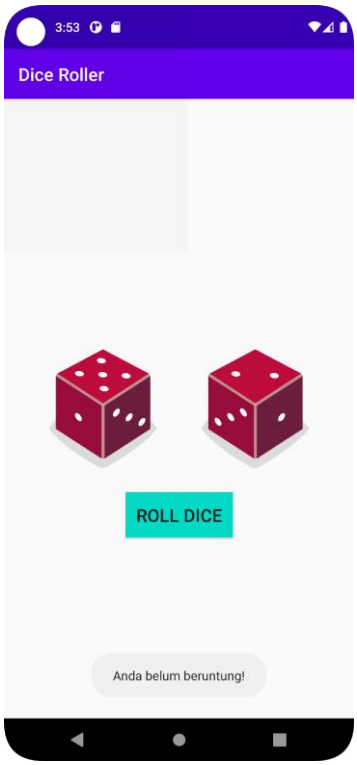
1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1.



Gambar 1 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.

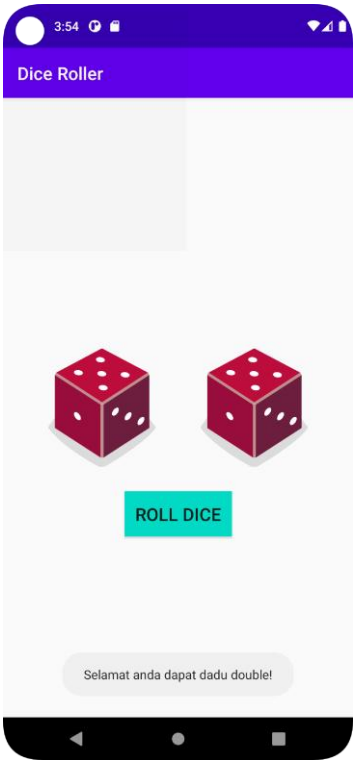


Gambar 2 Tampilan Dadu Setelah Di Roll

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.

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/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export= download



Gambar 3 Tampilan Roll Dadu Double

A. Source Code

MainActivity.kt

Tabel 1 Source Code Jawaban Soal 1

```
1  package com.example.diceroller
2
3  import android.os.Bundle
4  import android.widget.Button
5  import android.widget.ImageView
6  import android.widget.Toast
7  import androidx.appcompat.app.AppCompatActivity
8
   class MainActivity : AppCompatActivity() {
```

```
9       override fun onCreate(savedInstanceState: Bundle?) {
10          super.onCreate(savedInstanceState)
11          setContentView(R.layout.activity_main)
12
13          val diceImage01: ImageView =
    findViewById(R.id.imageView)
14          diceImage01.setImageResource(R.drawable.dice_0)
15
16          val diceImage02: ImageView =
    findViewById(R.id.imageView2)
17          diceImage02.setImageResource(R.drawable.dice_0)
18
19          val rollButton: Button = findViewById(R.id.button)
20
21          rollButton.setOnClickListener { rollDice() }
22      }
23
24      private fun rollDice() {
25          val dice1 = Dice(6)
26          val diceRoll1 = dice1.roll()
27
28          val diceImage1: ImageView =
    findViewById(R.id.imageView)
29          when (diceRoll1) {
30              1 -> diceImage1.setImageResource(R.drawable.dice_1)
31              2 -> diceImage1.setImageResource(R.drawable.dice_2)
32              3 -> diceImage1.setImageResource(R.drawable.dice_3)
33              4 -> diceImage1.setImageResource(R.drawable.dice_4)
34              5 -> diceImage1.setImageResource(R.drawable.dice_5)
35              6 -> diceImage1.setImageResource(R.drawable.dice_6)
36          }
37
38          val dice2 = Dice(6)
39          val diceRoll2 = dice2.roll()
40
41          val diceImage2: ImageView =
    findViewById(R.id.imageView2)
42          when (diceRoll2) {
43              1 -> diceImage2.setImageResource(R.drawable.dice_1)
44              2 -> diceImage2.setImageResource(R.drawable.dice_2)
45              3 -> diceImage2.setImageResource(R.drawable.dice_3)
46              4 -> diceImage2.setImageResource(R.drawable.dice_4)
47              5 -> diceImage2.setImageResource(R.drawable.dice_5)
48              6 -> diceImage2.setImageResource(R.drawable.dice_6)
49          }
50
```

```
51          if (diceRoll1 == diceRoll2){
52              val toast = Toast.makeText(this, "Selamat anda
    dapat dadu double!", Toast.LENGTH_SHORT)
53              toast.show()
54          } else {
55              val toast = Toast.makeText(this, "Anda belum
    beruntung!", Toast.LENGTH_SHORT)
56              toast.show() }
57
58      }
    }
59
60  class Dice(private val numSides: Int) {
61      fun roll(): Int {
62          return (1..numSides).random()
63      }
64  }
```

activity_main.xml

Tabel 2 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       android:layout_width="match_parent"
5       android:layout_height="match_parent"
6       xmlns:tools="http://schemas.android.com/tools"
7       xmlns:app="http://schemas.android.com/apk/res-auto"
8       tools:context="MainActivity">
9
10
11      <Button
12          android:id="@+id/button"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:layout_marginTop="52dp"
16          android:text="Dice Roll"
17          app:layout_constraintEnd_toEndOf="parent"
18          app:layout_constraintHorizontal_bias="0.498"
19          app:layout_constraintStart_toStartOf="parent"
20          app:layout_constraintTop_toBottomOf="@+id/imageView" />
21
22
23
24      <ImageView
```

```
25          android:id="@+id/imageView"
26          android:layout_width="150dp"
27          android:layout_height="150dp"
28          app:layout_constraintBottom_toBottomOf="parent"
29          app:layout_constraintEnd_toEndOf="parent"
30          app:layout_constraintHorizontal_bias="0.201"
31          app:layout_constraintStart_toStartOf="parent"
32          app:layout_constraintTop_toTopOf="parent"
33          app:layout_constraintVertical_bias="0.449"
34          app:srcCompat="@drawable/dice_1"
35          tools:layout_editor_absoluteX="83dp"
36          tools:layout_editor_absoluteY="268dp" />
37
38      <ImageView
39          android:id="@+id/imageView2"
40          android:layout_width="150dp"
41          android:layout_height="150dp"
42          app:layout_constraintBottom_toBottomOf="parent"
43          app:layout_constraintEnd_toEndOf="parent"
44          app:layout_constraintHorizontal_bias="0.837"
45          app:layout_constraintStart_toStartOf="parent"
46          app:layout_constraintTop_toTopOf="parent"
47          app:layout_constraintVertical_bias="0.449"
48          app:srcCompat="@drawable/dice_2" />
49
50
51  </androidx.constraintlayout.widget.ConstraintLayout>
```

B. Output Program

Gambar 4 Screenshot Output Jawaban Soal 1

C. Pembahasan

MainActivity.kt:

- **Pada Line 1,** package com.example.diceroller, Mendeklarasikan package tempat file Kotlin ini berada, yaitu com.example.diceroller.
- **Pada Line 3,** import android.os.Bundle, Mengimpor Bundle, yang digunakan untuk menyimpan data sementara antar aktivitas, terutama saat membuat Activity.
- **Pada Line 4,** import android.widget.Button, Mengimpor class Button untuk mengatur dan menangani tombol (button) dalam antarmuka pengguna.
- **Pada Line 5,** import android.widget.ImageView
  Mengimpor ImageView untuk menampilkan gambar (image) di dalam UI, seperti gambar dadu.
- **Pada Line 6,** import android.widget.Toast mengimpor Toast, yang digunakan untuk menampilkan pesan singkat (popup) kepada pengguna.
- **Pada Line 7,** import androidx.appcompat.app.AppCompatActivity
  Mengimpor AppCompatActivity, class dasar untuk Activity yang kompatibel ke versi Android lebih lama.
- **Pada Line 9,** class MainActivity : AppCompatActivity()
  Mendeklarasikan MainActivity yang merupakan turunan dari AppCompatActivity. Ini adalah entry point aplikasi saat dijalankan.
- **Pada Line 10-18,** Fungsi onCreate() dijalankan saat Activity dibuat.
    - **Pada Line 11,** super.onCreate(savedInstanceState)
      Memanggil implementasi onCreate dari superclass (AppCompatActivity).
    - **Pada Line 12,** setContentView(R.layout.activity_main)
      Mengatur layout XML activity_main.xml sebagai tampilan Activity ini.
    - **Pada Line 14,** val diceImage01: ImageView = findViewById(R.id.imageView) Menghubungkan komponen ImageView pertama dari XML ke variabel diceImage01.
    - **Pada Line 15,** diceImage01.setImageResource(R.drawable.dice_0)
      Menetapkan gambar awal (dice_0) ke ImageView pertama.

    - **Pada Line 17,** val diceImage02: ImageView = findViewById(R.id.imageView2) Menghubungkan ImageView kedua dari layout ke variabel diceImage02.
    - **Pada Line 18,**diceImage02.setImageResource(R.drawable.dice_0) Menetapkan gambar awal ke ImageView kedua.
    - **Pada Line 20,** val rollButton: Button = findViewById(R.id.button) Menghubungkan komponen tombol dari XML ke variabel rollButton.
    - **Pada Line 22,**rollButton.setOnClickListener { rollDice() } Menetapkan aksi saat tombol diklik, yaitu memanggil fungsi rollDice().
- **Pada Line 25,** val dice1 = Dice(6) Membuat objek dice1 dari class Dice dengan 6 sisi.
- **Pada Line 26,** val diceRoll1 = dice1.roll() Memanggil fungsi roll() untuk mendapatkan angka acak dari 1 sampai 6.
- **Pada Line 28,** val diceImage1: ImageView = findViewById(R.id.imageView) Menghubungkan kembali ImageView pertama untuk mengganti gambar sesuai hasil roll.
- **Pada Line 29-35,** Struktur when digunakan untuk mengganti gambar dadu berdasarkan nilai diceRoll1.
- **Pada Line 37,** val dice2 = Dice(6)
  Membuat objek dice2 untuk dadu kedua.
- **Pada Line 38,** val diceRoll2 = dice2.roll()
  Mengacak angka untuk dadu kedua.
- **Pada Line 40,** val diceImage2: ImageView = findViewById(R.id.imageView2)
  Menghubungkan ImageView kedua.
- **Pada Line 41-47,** Struktur when kedua untuk mengganti gambar dadu kedua berdasarkan hasil diceRoll2.
- **Pada Line 49-54,** Mengecek apakah kedua dadu menunjukkan angka yang sama:

    - Jika sama, tampilkan Toast “Selamat anda dapat dadu double!”
    - Jika berbeda, tampilkan Toast “Anda belum beruntung!”
- **Pada Line 57,** class Dice(private val numSides: Int) Mendefinisikan class Dice dengan parameter jumlah sisi numSides.
- **Pada Line 58-60,** Fungsi roll() menghasilkan angka acak dari 1 sampai jumlah sisi dadu menggunakan (1..numSides).random()

.

activity_main.xml

- Pada Line 1, <?xml version="1.0" encoding="utf-8"?> → Mendeklarasikan bahwa file ini adalah file XML dengan versi 1.0 dan encoding UTF-8.
- Pada Line 2, <androidx.constraintlayout.widget.ConstraintLayout ... >
  → Menggunakan ConstraintLayout sebagai layout utama untuk mengatur posisi elemen-elemen UI secara fleksibel dan responsif.
- Pada Line 3, xmlns:android="http://schemas.android.com/apk/res/android"
  → Mendefinisikan namespace standar Android untuk atribut seperti android:id, android:layout_width, dll.
- Pada Line 4, xmlns:tools="http://schemas.android.com/tools"
  → Namespace untuk atribut tools, hanya digunakan dalam Android Studio untuk pratinjau (tidak memengaruhi aplikasi saat dijalankan).
- Pada Line 5, xmlns:app="http://schemas.android.com/apk/res-auto"
  → Namespace untuk atribut khusus yang digunakan oleh pustaka AndroidX seperti ConstraintLayout (app:layout_constraint...).
- Pada Line 6, tools:context="MainActivity"
  → Menentukan konteks tampilan preview layout di Android Studio, yaitu activity MainActivity.
- Pada Line 9, <Button android:id="@+id/button" ... />
  → Membuat sebuah tombol dengan ID button yang bisa diakses di file Kotlin.

- Pada Line 10–11, android:layout_width="wrap_content" android:layout_height="wrap_content"
  → Ukuran tombol menyesuaikan dengan konten (teks) di dalamnya.
- Pada Line 12, android:layout_marginTop="52dp" → Menambahkan jarak 52dp dari atas ke tombol.
- Pada Line 13, android:text="Dice Roll" → Menampilkan teks “Dice Roll” pada tombol.
- Pada Line 14–17, → Mengatur posisi tombol agar berada di bawah imageView, dan berada di tengah horizontal menggunakan constraint:
  - app:layout_constraintTop_toBottomOf="@+id/imageView" → tombol berada di bawah gambar pertama.
  - app:layout_constraintStart_toStartOf="parent" dan app:layout_constraintEnd_toEndOf="parent" → mengatur agar tombol sejajar tengah horizontal.
  - app:layout_constraintHorizontal_bias="0.498" → sedikit menyesuaikan posisi horizontal.
- Pada Line 21 <ImageView android:id="@+id/imageView" ... /> → Gambar pertama dengan ID imageView (misal untuk dadu 1).
- Pada Line 22–23, android:layout_width="150dp" android:layout_height="150dp"
  → Menentukan ukuran gambar sebesar 150dp x 150dp.
- Pada Line 24–30, → Mengatur gambar agar berada di tengah vertikal dan horizontal menggunakan constraint:
  - app:layout_constraintTop_toTopOf="parent" dan app:layout_constraintBottom_toBottomOf="parent" → menempatkan gambar di tengah secara vertikal.

  - app:layout_constraintStart_toStartOf="parent" dan app:layout_constraintEnd_toEndOf="parent" → menempatkan gambar di tengah secara horizontal.
  - app:layout_constraintHorizontal_bias="0.201" → menggeser gambar ke kiri.
- Pada Line 31, app:srcCompat="@drawable/dice_1"→ Menampilkan gambar dice_1 dari folder res/drawable.
- Pada Line 32–33, tools:layout_editor_absoluteX dan tools:layout_editor_absoluteY → Hanya untuk tampilan editor Android Studio, tidak berpengaruh saat dijalankan.
- Pada Line 36, <ImageView android:id="@+id/imageView2" ... /> → Gambar kedua dengan ID imageView2 (misal untuk dadu 2).
- Pada Line 37–38, android:layout_width="150dp" android:layout_height="150dp" → Ukuran gambar 150dp x 150dp.
- Pada Line 39–45, → Posisi gambar hampir sama seperti imageView, namun app:layout_constraintHorizontal_bias="0.837" menggeser gambar ke kanan.
- Pada Line 46, app:srcCompat="@drawable/dice_2" → Menampilkan gambar dice_2 dari folder res/drawable.
- Pada Line 49, </androidx.constraintlayout.widget.ConstraintLayout> → Menutup tag ConstraintLayout, mengakhiri deklarasi layout.

.

MODUL 2 : ANDROID LAYOUT

SOAL 1

Soal Praktikum:

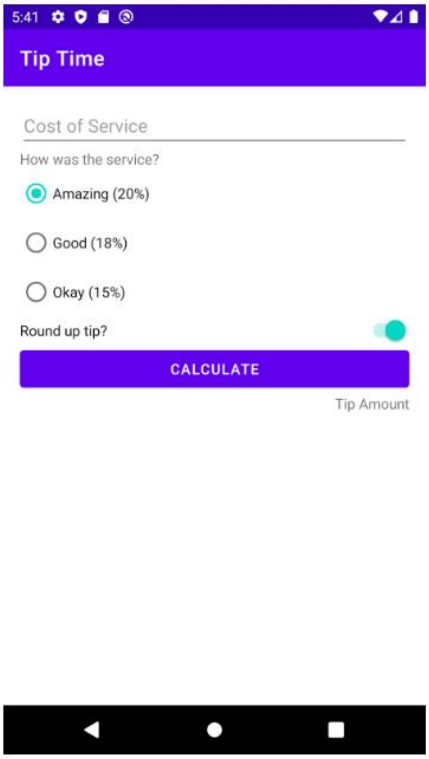
Buatlah sebuah aplikasi kalkulator tip yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

1. Input Biaya Layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.

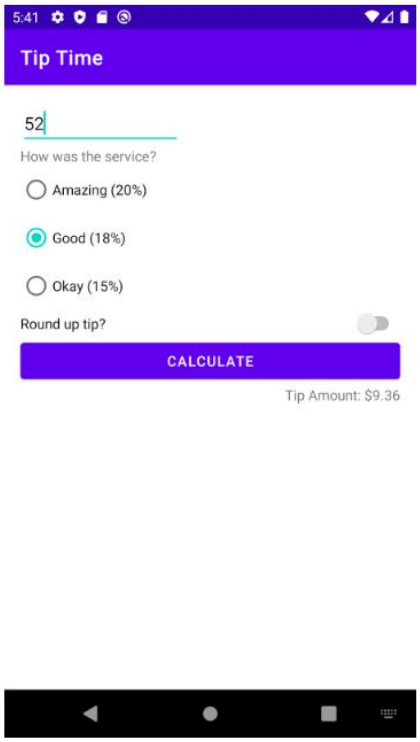
2. Pilihan Persentase Tip: Pengguna dapat memilih persentase tip yang diinginkan dari opsi yang disediakan, yaitu 15%, 18%, dan 20%.

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 5 Tampilan Awal Aplikasi



Gambar 6 Tampilan Aplikasi Setelah Dijalankan

A. Source Code

MainActivity.kt

Tabel 3 Source Code Jawaban Soal 1

```
1   package com.example.tipkalkulator
2
3   import android.os.Bundle
4   import android.widget.*
5   import androidx.appcompat.app.AppCompatActivity
6   import java.text.NumberFormat
7   import kotlin.math.ceil
8
9   class MainActivity : AppCompatActivity() {
10
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          setContentView(R.layout.activity_main)
14
```

```
15          val calcButton: Button =
16  findViewById(R.id.calculate_button)
17          calcButton.setOnClickListener { calcTip() }
18      }
19
20      private fun calcTip() {
21          val costEditText: EditText =
22  findViewById(R.id.cost_of_service_edit_text)
23          val costString = costEditText.text.toString()
24
25          if (costString.isEmpty()) {
26              displayTip(0.0)
27              return
28          }
29
30          val cost = costString.toDoubleOrNull()
31          if (cost == null || cost == 0.0) {
32              displayTip(0.0)
33              return
34          }
35
36          val tipPercent = when
37  (findViewById<RadioGroup>(R.id.tip_options).checkedRadioButtonId)
38  {
39              R.id.amazing_option -> 0.20
40              R.id.good_option -> 0.18
41              else -> 0.15
42          }
43
44          var tip = tipPercent * cost
45          val roundUp =
46  findViewById<Switch>(R.id.round_up_switch).isChecked
47          if (roundUp) {
48              tip = ceil(tip)
49          }
50
51          displayTip(tip)
52      }
53
54      private fun displayTip(tip: Double) {
55          val formattedTip =
56  NumberFormat.getCurrencyInstance().format(tip)
57          val tipResult: TextView = findViewById(R.id.tip_result)
58          tipResult.text = getString(R.string.tip_amount,
59  formattedTip)
60      }
```

```
61  }
```

activity_main.xml

Tabel 4 Source Code Jawaban Soal 1

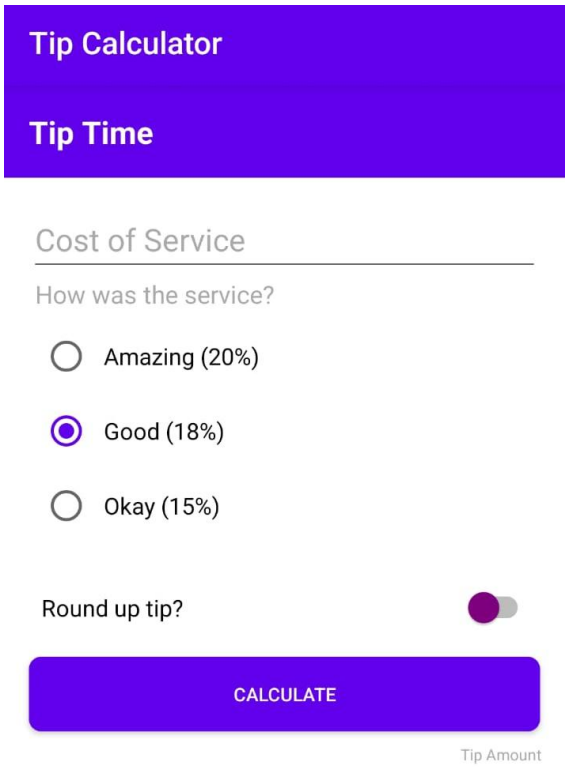
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:tools="http://schemas.android.com/tools"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       android:background="@color/white"
9       tools:context=".MainActivity">
10
11      <TextView
12          android:id="@+id/app_title"
13          android:layout_width="0dp"
14          android:layout_height="wrap_content"
15          android:text="Tip Time"
16          android:textColor="@android:color/white"
17          android:textSize="20sp"
18          android:textStyle="bold"
19          android:background="#6200ee"
20          android:padding="16dp"
21          app:layout_constraintTop_toTopOf="parent"
22          app:layout_constraintStart_toStartOf="parent"
23          app:layout_constraintEnd_toEndOf="parent" />
24
25      <EditText
26          android:id="@+id/cost_of_service_edit_text"
27          android:layout_width="0dp"
28          android:layout_height="wrap_content"
29          android:layout_margin="16dp"
30          android:hint="Cost of Service"
31          android:inputType="numberDecimal"
32          android:textSize="20sp"
33          android:textColorHint="@android:color/darker_gray"
34          app:layout_constraintTop_toBottomOf="@id/app_title"
35          app:layout_constraintStart_toStartOf="parent"
36          app:layout_constraintEnd_toEndOf="parent" />
37
38      <TextView
39          android:id="@+id/tip"
40          android:layout_width="wrap_content"
41          android:layout_height="wrap_content"
42          android:layout_marginStart="20dp"
```

```
43          android:layout_marginTop="0dp"
44          android:text="How was the service?"
45          android:textColor="@android:color/darker_gray"
46          android:textSize="16sp"
47
48  app:layout_constraintTop_toBottomOf="@id/cost_of_service_edit_text"
49          app:layout_constraintStart_toStartOf="parent"/>
50
51      <RadioGroup
52          android:id="@+id/tip_options"
53          android:layout_width="wrap_content"
54          android:layout_height="wrap_content"
55          android:layout_marginStart="24dp"
56          android:layout_marginTop="5dp"
57          app:layout_constraintTop_toBottomOf="@id/tip"
58          app:layout_constraintStart_toStartOf="parent">
59
60          <RadioButton
61              android:id="@+id/amazing_option"
62              android:layout_width="wrap_content"
63              android:layout_height="wrap_content"
64              android:text="Amazing (20%)"
65              android:textColor="@android:color/black"
66              android:textSize="15sp"
67              android:checked="true"
68              android:minHeight="48dp"
69              android:padding="8dp" />
70
71          <RadioButton
72              android:id="@+id/good_option"
73              android:layout_width="wrap_content"
74              android:layout_height="wrap_content"
75              android:text="Good (18%)"
76              android:textColor="@android:color/black"
77              android:textSize="15sp"
78              android:minHeight="48dp"
79              android:padding="8dp" />
80
81          <RadioButton
82              android:id="@+id/okay_option"
83              android:layout_width="wrap_content"
84              android:layout_height="wrap_content"
85              android:text="Okay (15%)"
86              android:textColor="@android:color/black"
87              android:textSize="15sp"
88              android:minHeight="48dp"
```

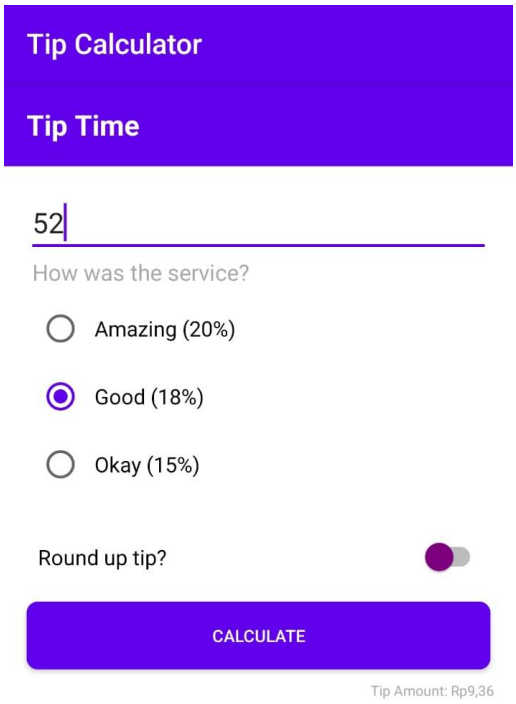
```
89              android:padding="8dp" />
90      </RadioGroup>
91
92      <Switch
93          android:id="@+id/round_up_switch"
94          android:layout_width="317dp"
95          android:layout_height="53dp"
96          android:layout_marginStart="24dp"
97          android:layout_marginTop="15dp"
98          android:text="Round up tip?"
99          android:textColor="@android:color/black"
100         android:textSize="15sp"
101         android:thumbTint="#800080"
102         app:layout_constraintStart_toStartOf="parent"
103         app:layout_constraintTop_toBottomOf="@id/tip_options"
104         tools:ignore="HardcodedText,UseSwitchCompatOrMaterialXml"
105 />
106
107     <Button
108         android:id="@+id/calculate_button"
109         android:layout_width="0dp"
110         android:layout_height="48dp"
111         android:layout_marginTop="5dp"
112         android:text="CALCULATE"
113         android:textColor="@color/white"
114         android:textSize="12sp"
115         android:gravity="center"
116         android:background="@drawable/button"
117         app:layout_constraintTop_toBottomOf="@id/round_up_switch"
118         app:layout_constraintStart_toStartOf="parent"
119         app:layout_constraintEnd_toEndOf="parent"
120         android:layout_marginStart="16dp"
121         android:layout_marginEnd="16dp"
122         tools:ignore="HardcodedText" />
123
124     <TextView
125         android:id="@+id/tip_result"
126         android:layout_width="wrap_content"
127         android:layout_height="wrap_content"
128         android:layout_marginTop="8dp"
129         android:text="Tip Amount"
130         android:textColor="@android:color/darker_gray"
131         android:textSize="10sp"
132         app:layout_constraintEnd_toEndOf="parent"
133         app:layout_constraintHorizontal_bias="0.951"
134         app:layout_constraintStart_toStartOf="parent"
```

```
135         app:layout_constraintTop_toBottomOf="@id/calculate_button"
136 />
137
138 </androidx.constraintlayout.widget.ConstraintLayout>
```

B. Output Program



Gambar 7 Hasil Tampilan Awal Aplikasi Soal 1



Gambar 8 Hasil Tampilan Aplikasi Setelah Dijalankan Soal 1

C. Pembahasan

MainActivity.kt:

• Pada Line 1, package com.example.tipkalkulator Mendeklarasikan package tempat file Kotlin ini berada, yaitu com.example.tipkalkulator.

• Pada Line 3, import android.os.Bundle Mengimpor Bundle, yang digunakan untuk menyimpan data sementara saat activity dijalankan.

• Pada Line 4, import android.widget.* Mengimpor semua komponen UI seperti Button, EditText, TextView, RadioGroup, Switch, dll.

• Pada Line 5, import androidx.appcompat.app.AppCompatActivityMengimpor AppCompatActivity, class dasar untuk activity dengan kompatibilitas Android yang lebih luas.
• Pada Line 6, import java.text.NumberFormat Digunakan untuk memformat angka menjadi format mata uang (currency).
• Pada Line 7, import kotlin.math.ceil Mengimpor fungsi ceil() dari Kotlin untuk membulatkan angka ke atas.
• Pada Line 9, class MainActivity : AppCompatActivity() Mendeklarasikan class utama aplikasi yang merupakan turunan dari AppCompatActivity.
• Pada Line 11, override fun onCreate(savedInstanceState: Bundle?) Method yang dijalankan saat activity dibuat pertama kali.
• Pada Line 12, setContentView(R.layout.activity_main) Mengatur tampilan activity berdasarkan file layout activity_main.xml.
• Pada Line 14, val calcButton: Button = findViewById(R.id.calculate_button) Mencari tombol berdasarkan ID-nya dan menyimpannya ke dalam variabel calcButton.
• Pada Line 15, calcButton.setOnClickListener { calcTip() }Menambahkan aksi klik tombol yang akan memanggil fungsi calcTip() saat ditekan.
• Pada Line 18, fun calcTip() Fungsi untuk menghitung tip berdasarkan input pengguna.
• Pada Line 19, val costEditText: EditText = findViewById(R.id.cost_of_service_edit_text) Mengambil input dari pengguna terkait biaya layanan.
• Pada Line 20, val costString = costEditText.text.toString() Mengonversi isi EditText menjadi string.
• Pada Line 22-25, Mengecek apakah input kosong atau bernilai nol, jika iya maka tip ditampilkan sebagai 0.0 dan fungsi dihentikan.
• Pada Line 27, val tipPercent = when (...) Menentukan persentase tip berdasarkan pilihan radio button (Amazing, Good, Okay).

• Pada Line 32, var tip = tipPercent * cost Menghitung jumlah tip berdasarkan biaya layanan dan persentase tip.
• Pada Line 33, val roundUp = findViewById<Switch>(R.id.round_up_switch).isChecked Mengambil status dari switch "Round up tip?"
• Pada Line 34-36, Jika roundUp aktif, nilai tip dibulatkan ke atas menggunakan ceil().
• Pada Line 38, displayTip(tip) Memanggil fungsi untuk menampilkan hasil tip.
• Pada Line 41, fun displayTip(tip: Double) Fungsi untuk menampilkan nilai tip dalam format mata uang.
• Pada Line 42, val formattedTip = umberFormat.getCurrencyInstance().format(tip)
Memformat nilai tip menjadi format uang (misalnya Rp, $, dll).
• Pada Line 43, val tipResult: TextView = findViewById(R.id.tip_result)
Mengambil TextView tempat hasil tip akan ditampilkan.
• Pada Line 44, tipResult.text = getString(R.string.tip_amount, formattedTip)
Menampilkan hasil tip dalam TextView.

.

activity_main.xml

• Pada Line 1, <?xml version="1.0" encoding="utf-8"?> Mendefinisikan versi XML yang digunakan serta format encoding-nya, yaitu UTF-8.

• Pada Line 2, <androidx.constraintlayout.widget.ConstraintLayout ...> Merupakan root layout yang digunakan untuk mengatur tata letak UI dengan sistem constraint.

• Pada Line 3-5, xmlns:android, xmlns:tools, xmlns:app Mendefinisikan namespace yang diperlukan untuk atribut standar Android, tools dari Android Studio, dan fitur khusus AndroidX.

• Pada Line 6-8, android:layout_width="match_parent" dan android:layout_height="match_parent"

Menentukan ukuran layout agar memenuhi seluruh layar.

• Pada Line 9, android:background="@color/white"

Mengatur latar belakang layout menjadi putih.

• Pada Line 10, tools:context=".MainActivity"

Digunakan oleh Android Studio untuk preview layout dalam konteks MainActivity.

• Pada Line 12-20, <TextView android:id="@+id/app_title" ...>

Membuat judul aplikasi "Tip Time" di bagian atas layar dengan warna latar ungu dan teks putih.

• Pada Line 22-30, <EditText android:id="@+id/cost_of_service_edit_text" ...>

Input teks tempat pengguna mengisi biaya layanan yang akan dihitung tip-nya. Bertipe numerik desimal.

• Pada Line 32-38, <TextView android:id="@+id/tip" ...>

Label untuk pertanyaan “How was the service?” sebagai petunjuk sebelum memilih opsi tip.

• Pada Line 40-56, <RadioGroup android:id="@+id/tip_options" ...>

Kumpulan pilihan tip berdasarkan kualitas layanan. Terdiri dari tiga RadioButton:

- Pada Line 42-47, <RadioButton android:id="@+id/amazing_option" ...>
  Pilihan tip 20% untuk layanan "Amazing". Sudah tercentang secara default.
- Pada Line 48-52, <RadioButton android:id="@+id/good_option" ...>
  Pilihan tip 18% untuk layanan "Good".
- Pada Line 53-56, <RadioButton android:id="@+id/okay_option" ...>
  Pilihan tip 15% untuk layanan "Okay".
- Pada Line 58-64, <Switch android:id="@+id/round_up_switch" ...>
  Switch untuk memilih apakah tip ingin dibulatkan ke atas atau tidak.
- Pada Line 66-75, <Button android:id="@+id/calculate_button" ...>
  Tombol untuk memulai perhitungan tip berdasarkan input dan pilihan pengguna.

- Pada Line 77-83, <TextView android:id="@+id/tip_result" ...>
Menampilkan hasil dari perhitungan tip dalam bentuk format mata uang.

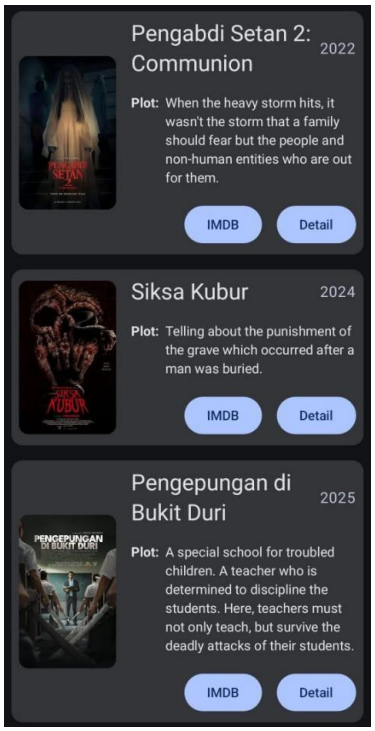
MODUL 3 : Build a scrollable list

SOAL 1

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

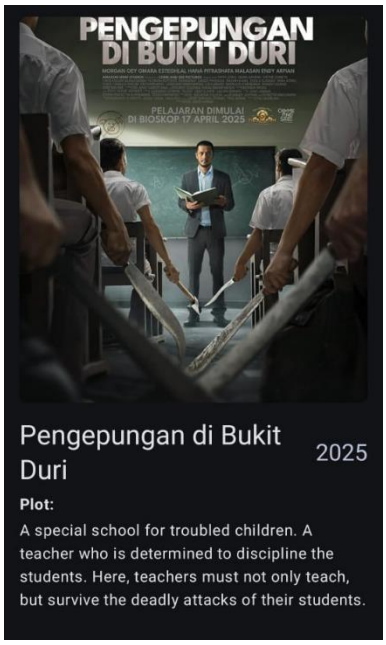
1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah
4. Terdapat 2 button dalam list, dengan fungsi berikut:
    a. Button pertama 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 9 Contoh UI list

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 10 Contoh UI Detail

A. Source Code

1. MainActivity.kt

Tabel 5 Source Code Jawaban Soal 1

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

2. DetailFragment.kt

Tabel 6 Source Code Jawaban Soal 1

```
1   package com.example.ootdlist.fragments
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
```

```
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import androidx.fragment.app.Fragment
10  import com.example.ootdlist.data.Ootd
11  import com.example.ootdlist.databinding.FragmentDetailBinding
12  import java.net.URLEncoder
13
14  class DetailFragment : Fragment() {
15      private var _binding: FragmentDetailBinding? = null
16      private val binding get() = _binding!!
17
18      override fun onCreateView(
19          inflater: LayoutInflater,
20          container: ViewGroup?,
21          savedInstanceState: Bundle?
22      ): View {
23          _binding = FragmentDetailBinding.inflate(inflater,
    container, false)
24          return binding.root
25      }
26
27      override fun onViewCreated(view: View, savedInstanceState:
    Bundle?) {
28          super.onViewCreated(view, savedInstanceState)
29
30          val ootdId = requireArguments().getInt("ootdId")
31          val ootd = Ootd.getById(ootdId)
32
33          if (ootd != null) {
34              binding.ootdTitle.text = ootd.title
35              binding.ootdGender.text = ootd.gender
36              binding.ootdStyle.text = ootd.style
37              binding.ootdDescription.text = ootd.description
38              binding.ootdImage.setImageResource(ootd.imageResId)
39
40              binding.btnShop.setOnClickListener {
41                  val searchQuery = "${ootd.title} ${ootd.style}
    ${ootd.gender} fashion"
42                  val encodedQuery =
    URLEncoder.encode(searchQuery, "UTF-8")
43                  val url =
    "https://shopee.co.id/search?keyword=$encodedQuery"
44                  val intent = Intent(Intent.ACTION_VIEW,
    Uri.parse(url))
45                  startActivity(intent)
46              }
47          }
48      }
```

```
49
50      override fun onDestroyView() {
51          super.onDestroyView()
52          _binding = null
53      }
54  }
55
```

3. ListFragment.kt

Tabel 7 Source Code Jawaban Soal 1

```
1   package com.example.ootdlist.fragments
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import androidx.fragment.app.Fragment
10  import androidx.navigation.fragment.findNavController
11  import androidx.recyclerview.widget.LinearLayoutManager
12  import com.example.ootdlist.R
13  import com.example.ootdlist.adapter.OotdAdapter
14  import com.example.ootdlist.data.Ootd
15  import com.example.ootdlist.databinding.FragmentListBinding
16  import java.net.URLEncoder
17
18  class ListFragment : Fragment() {
19      private var _binding: FragmentListBinding? = null
20      private val binding get() = _binding!!
21
22      override fun onCreateView(
23          inflater: LayoutInflater,
24          container: ViewGroup?,
25          savedInstanceState: Bundle?
26      ): View {
27          _binding = FragmentListBinding.inflate(inflater,
    container, false)
28          return binding.root
29      }
30
31      override fun onViewCreated(view: View, savedInstanceState:
32  Bundle?) {
33          super.onViewCreated(view, savedInstanceState)
```

```
34
35          val ootdList = Ootd.ootdList
36          val adapter = OotdAdapter(ootdList, object :
    OotdAdapter.OnItemClickListener {
37              override fun onDetailClick(ootd: Ootd) {
38                  val bundle = Bundle().apply {
39                      putInt("ootdId", ootd.id)
40                  }
41
    findNavController().navigate(R.id.action_listFragment_to_detailFr
    agment, bundle)
42              }
43
44              override fun onShopClick(ootd: Ootd) {
45                  val searchQuery = "${ootd.title} ${ootd.style}
46  ${ootd.gender} fashion"
                    val encodedQuery = URLEncoder.encode(searchQuery,
47  "UTF-8")
                    val url =
48  "https://shopee.co.id/search?keyword=$encodedQuery"
                    val intent = Intent(Intent.ACTION_VIEW,
49  Uri.parse(url))
50                  startActivity(intent)
51              }
52          })
53
54          binding.recyclerView.layoutManager =
    LinearLayoutManager(context)
55          binding.recyclerView.adapter = adapter
56
57          if (ootdList.isEmpty()) {
58              binding.recyclerView.visibility = View.GONE
59              binding.emptyView.visibility = View.VISIBLE
60          } else {
61              binding.recyclerView.visibility = View.VISIBLE
62              binding.emptyView.visibility = View.GONE
63          }
64      }
65
66      override fun onDestroyView() {
67          super.onDestroyView()
68          _binding = null
69      }
    }
```

4. OotdAdapter.kt

Tabel 8 Source Code Jawaban Soal 1

```
1   package com.example.ootdlist.adapter
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.RecyclerView
6   import com.example.ootdlist.R
7   import com.example.ootdlist.data.Ootd
8   import com.example.ootdlist.databinding.ItemOotdBinding
9
10  class OotdAdapter(private val ootdList: List<Ootd>, private val
    listener: OnItemClickListener) :
11      RecyclerView.Adapter<OotdAdapter.OotdViewHolder>() {
12
13      interface OnItemClickListener {
14          fun onDetailClick(ootd: Ootd)
15          fun onShopClick(ootd: Ootd)
16      }
17
18      inner class OotdViewHolder(private val binding:
19  ItemOotdBinding) : RecyclerView.ViewHolder(binding.root) {
            fun bind(ootd: Ootd) {
20              binding.ootdTitle.text = ootd.title
21              binding.ootdGender.text = ootd.gender
22              binding.ootdStyle.text = ootd.style
23              binding.ootdDescription.text = ootd.description
24              binding.ootdImage.setImageResource(ootd.imageResId)
25
26              binding.btnImdb.setOnClickListener {
27                  listener.onShopClick(ootd)
28              }
29
30              binding.btnDetail.setOnClickListener {
31                  listener.onDetailClick(ootd)
32              }
33
34              val imageContentDesc = itemView.context.getString(
35                  R.string.ootd_image_content_description,
36                  ootd.title,
37                  ootd.gender
38              )
39              binding.ootdImage.contentDescription =
```

```
40  imageContentDesc
            }
41      }
42
43      override fun onCreateViewHolder(parent: ViewGroup, viewType:
44  Int): OotdViewHolder {
            val binding =
45  ItemOotdBinding.inflate(LayoutInflater.from(parent.context),
    parent, false)
            return OotdViewHolder(binding)
46      }
47
48      override fun onBindViewHolder(holder: OotdViewHolder,
49  position: Int) {
            holder.bind(ootdList[position])
50      }
51
52      override fun getItemCount(): Int = ootdList.size
53  }
```

5. OotdData.kt

Tabel 9 Source Code Jawaban Soal 1

```
1   package com.example.ootdlist.data
2
3   import com.example.ootdlist.R
4
5   data class Ootd(
6       val id: Int,
7       val title: String,
8       val style: String,
9       val description: String,
10      val imageResId: Int,
11      val gender: String
12  ) {
13      companion object {
14          val ootdList = listOf(
15              Ootd(1, "Block Core", "Streetwear", "gaya streetwear
    modern yang lahir dari semangat bereksperimen dan mengaburkan
    batas antara dunia olahraga dan fashion jalanan. " +
16                      "Keunikannya terletak pada penggunaan jersey
    bola sebagai item utama, menjadikannya lebih dari sekadar simbol
```

```
        fandom—melainkan pernyataan gaya yang berani. " +
17                      "Gaya ini merepresentasikan generasi muda
    yang dinamis, penuh karakter, dan tidak takut memadukan elemen
    yang tampak kontras. Block Core bukan hanya soal pakaian," +
18                      " tapi juga tentang cara mengekspresikan
    diri dengan bebas dan orisinal." +
19                      "\n\n Atasan: Jersey klub bola bergaya retro
    dengan motif kotak merah-putih dan logo “PACHA Ibiza” yang
    menonjol." +
20                      "\n\n Bawahan: Celana wide leg bahan wol
    berwarna abu-abu tua, menciptakan kontras menarik antara sporty
    dan formal.", R.drawable.block_core_man, "Men"),
21
22              Ootd(2, "Block Core", "Streetwear", " gaya
    streetwear modern yang lahir dari semangat bereksperimen dan
    mengaburkan batas antara dunia olahraga dan fashion jalanan. " +
23                      "Keunikannya terletak pada penggunaan jersey
    bola sebagai item utama, menjadikannya lebih dari sekadar simbol
    fandom—melainkan pernyataan gaya yang berani. " +
24                      "Gaya ini merepresentasikan generasi muda
    yang dinamis, penuh karakter, dan tidak takut memadukan elemen
    yang tampak kontras. Block Core bukan hanya soal pakaian, " +
25                      "tapi juga tentang cara mengekspresikan diri
    dengan bebas dan orisinal." +
26                      "\n\n Atasan: Jersey FC Barcelona dengan
    potongan crop top, menampilkan logo Nike dan sponsor Spotify
    yang mencolok." +
27                      "\n\n Bawahan: Celana cargo hitam longgar,
    menciptakan siluet santai namun tetap bold.",
    R.drawable.block_core_girl, "Women"),
28
29              Ootd(3, "Cottage Core", "Romantic", "gaya fashion
    yang terinspirasi dari kehidupan pedesaan yang tenang, romantis,
    dan penuh nostalgia. " +
30                      "Gaya ini mengangkat elemen alami dan
    vintage—mulai dari motif bunga kecil, bahan katun atau linen,
    warna-warna hangat dan lembut," +
31                      " hingga siluet yang mengalir dan nyaman.
    Cottage Core bukan hanya tentang penampilan, tetapi juga
    menggambarkan gaya hidup yang menghargai" +
32                      " keindahan sederhana, kedekatan dengan
    alam, dan suasana yang damai." +
33                      "\n\n Atasan: Sweater rajut motif folk
    berwarna dasar hitam dengan aksen pola merah-putih yang memberi
    kesan hangat." +
34                      "\n\n Bawahan: Celana panjang corduroy
```

```
        coklat muda berpotongan longgar, menambah kesan earthy dan
        santai.", R.drawable.cottage_core_man, "Men"),
35
36              Ootd(4, "Cottage Core", "Romantic", "gaya fashion
        yang terinspirasi dari kehidupan pedesaan yang tenang, romantis,
        dan penuh nostalgia." +
37                      " Gaya ini mengangkat elemen alami dan
        vintage—mulai dari motif bunga kecil, bahan katun atau linen,
        warna-warna hangat dan lembut," +
38                      " hingga siluet yang mengalir dan nyaman.
        Cottage Core bukan hanya tentang penampilan, tetapi juga
        menggambarkan gaya hidup yang menghargai " +
39                      "keindahan sederhana, kedekatan dengan alam,
        dan suasana yang damai." +
40                      "\n\n Atasan: Kardigan rajut tipis berwarna
        merah dipadukan dengan dress floral putih bermotif bunga kecil
        dan detail renda." +
41                      "\n\n Bawahan: Dress panjang mengalir dengan
        potongan klasik dan aksen pita, menciptakan siluet lembut yang
        feminin.", R.drawable.cottage_core_girl, "Women"),
42
43              Ootd(5, "Grunge", "Alternative", "Outfit bergaya
        grunge identik dengan kesan santai, berantakan yang terkesan
        effortless namun tetap keren. " +
44                      "Ciri khasnya adalah dominasi warna gelap,
        layering, bahan denim, serta kaos band sebagai elemen penting. "
        +
45                      "Gaya ini memadukan unsur musik alternatif
        dengan sikap anti-mainstream khas era 90-an." +
46                      "\n\n Atasan: Kaos grafis berwarna hitam
        dilapisi kemeja flanel motif kotak merah-hijau yang terbuka,
        menciptakan tampilan layering khas grunge klasik." +
47                      "\n\n Bawahan: Celana longgar berbahan
        corduroy hijau tua, memberi tekstur earthy yang sejalan dengan
        elemen grunge vintage.", R.drawable.grunge_man, "Men"),
48
49              Ootd(6, "Grunge", "Alternative", "Outfit bergaya
        grunge identik dengan kesan santai, berantakan yang terkesan
        effortless namun tetap keren. " +
50                      "Ciri khasnya adalah dominasi warna gelap,
        layering, bahan denim, serta kaos band sebagai elemen penting."
        +
51                      " Gaya ini memadukan unsur musik alternatif
        dengan sikap anti-mainstream khas era 90-an." +
52                      "\n\n Atasan: Kaos band The Beatles berwarna
        hitam kusam dengan print besar, memberikan nuansa retro dan
```

```
        rebellious." +
53                      "\n\n Bawahan: Celana jeans baggy
        berpotongan longgar berwarna biru pudar, menambah kesan santai
        dan nonchalant.", R.drawable.grunge_girl, "Women"),
54
55              Ootd(7, "Japanese Americana", "Fusion", "Outfit
        bergaya Japanese Americana memadukan gaya kasual klasik khas
        Amerika seperti denim dan workwear dengan sentuhan rapi,
        minimalis, dan estetika layering ala Jepang. " +
56                      "Ciri khasnya meliputi jaket denim, celana
        berpotongan longgar, warna earthy seperti olive dan navy, serta
        perhatian terhadap siluet dan tekstur." +
57                      "\n\n Atasan: Kemeja putih berkerah dengan
        dasi cokelat yang dilapisi jaket denim fitted berwarna indigo,
        menciptakan tampilan rapi dengan lapisan Americana yang kuat." +
58                      "\n\n Bawahan: Celana workwear hijau zaitun
        berpotongan lurus yang terinspirasi dari gaya militer,
        menyeimbangkan kesan maskulin dan vintage.",
        R.drawable.japanese_americana_man, "Men"),
59
60              Ootd(8, "Japanese Americana", "Fusion", "Outfit
        bergaya Japanese Americana memadukan gaya kasual klasik khas
        Amerika seperti denim dan workwear dengan sentuhan rapi,
        minimalis, dan estetika layering ala Jepang. " +
61                      "Ciri khasnya meliputi jaket denim, celana
        berpotongan longgar, warna earthy seperti olive dan navy, serta
        perhatian terhadap siluet dan tekstur." +
62                      "\n\n Atasan: Kaos bergaris hitam-putih
        bergaya Breton yang dimasukkan ke dalam celana, memberikan kesan
        clean namun santai. Sebuah jaket denim biru dikenakan secara
        kasual di bahu sebagai outer, menambah sentuhan Americana
        klasik." +
63                      "\n\n Bawahan: Celana cargo hijau army
        berpotongan lebar yang memberi nuansa utilitarian khas workwear,
        sekaligus menciptakan siluet longgar yang modis dan nyaman.",
        R.drawable.japanese_americana_girl, "Women"),
64
65              Ootd(9, "Old Money", "Classic", "Outfit bergaya Old
        Money mencerminkan kemewahan yang tidak mencolok, elegan, dan
        timeless. Gaya ini lekat dengan nuansa klasik, potongan yang
        rapi," +
66                      " bahan berkualitas tinggi, serta warna-
        warna netral yang terkesan mahal. Tidak berfokus pada tren,
        tetapi pada keanggunan dan prestise." +
67                      "\n\n Atasan: Kemeja putih klasik dengan
        kerah terbuka, memberikan kesan effortless dan bersih namun
```

```
        tetap terjaga rapi. Diselipkan ke dalam celana untuk tampilan
        yang lebih polished." +
68                      "\n\n Bawahan: Celana wool cokelat
        berpotongan longgar dan pleated, menambah kesan sophisticated
        khas gentleman lama. ", R.drawable.old_money_man, "Men"),
69
70              Ootd(10, "Old Money", "Classic", "Outfit bergaya Old
        Money mencerminkan kemewahan yang tidak mencolok, elegan, dan
        timeless. Gaya ini lekat dengan nuansa klasik, potongan yang
        rapi," +
71                      " bahan berkualitas tinggi, serta warna-
        warna netral yang terkesan mahal. Tidak berfokus pada tren,
        tetapi pada keanggunan dan prestise." +
72                      "\n\n Atasan: Kemeja satin berwarna ivory
        dengan detail kerah klasik, memberikan tampilan sophisticated
        yang mewah namun understated. Dipadukan dengan blazer oversized
        dalam nuansa beige yang memberikan siluet structured dan
        elegan." +
73                      "\n\n Bawahan: Celana tailored high-waist
        warna senada yang jatuh longgar hingga menyentuh lantai,
        menegaskan citra refined dan berkelas.",
        R.drawable.old_money_girl, "Women"),
74
75              Ootd(11, "Y2K", "Retro Futurism", "Outfit bergaya
        Y2K (Year 2000) identik dengan nuansa futuristik, playful, dan
        eksperimental yang merefleksikan semangat awal milenium. " +
76                      "Gaya ini sering menggabungkan elemen-elemen
        seperti low-rise jeans, atasan ketat, bahan denim dengan efek
        faded, " +
77                      "serta aksesori mencolok yang terinspirasi
        dari dunia pop culture, teknologi, dan fashion selebriti tahun
        2000-an." +
78                      "\n\n Atasan: Kemeja putih sebagai dasar
        dipadukan dengan sweater hitam dan jaket kulit hitam sebagai
        outer, menghasilkan tampilan layering maskulin dan berani yang
        populer di era awal 2000-an." +
79                      "\n\n Bawahan: Celana jeans longgar berwarna
        hitam pudar, menciptakan kesan laid-back namun tetap tajam,
        selaras dengan gaya street style Y2K.", R.drawable.y2k_man,
        "Men"),
80
81              Ootd(12, "Y2K", "Retro Futurism", "Outfit bergaya
        Y2K (Year 2000) identik dengan nuansa futuristik, playful, dan
        eksperimental yang merefleksikan semangat awal milenium. " +
82                      "Gaya ini sering menggabungkan elemen-elemen
        seperti low-rise jeans, atasan ketat, bahan denim dengan efek
```

```
      faded, " +
83                    "serta aksesori mencolok yang terinspirasi
      dari dunia pop culture, teknologi, dan fashion selebriti tahun
      2000-an." +
84                    "\n\n Atasan: Cropped cardigan abu-abu tua
      yang ketat dipadukan dengan tank top abu-abu muda, menciptakan
      tampilan layering yang menggoda dan edgy." +
85                    "\n\n Bawahan: Celana jeans flare
      berpotongan low-rise dengan efek washed-out, memperkuat siluet
      tubuh dan nuansa retro futuristik khas Y2K.",
      R.drawable.y2k_girl, "Women")
86            )
87
88            fun getById(id: Int): Ootd? = ootdList.find { it.id ==
      id }
89        }
90    }
```

6. activity_main.xml

Tabel 10 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph" />
</FrameLayout>
```

7. Fragment_detail.xml

Tabel 11 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    android:padding="16dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/ootdImage"
            android:layout_width="match_parent"
            android:layout_height="500dp"
            android:background="@color/gray_300"
```

```
18              android:scaleType="centerCrop"
19              app:layout_constraintTop_toTopOf="parent" />
20
21          <TextView
22              android:id="@+id/ootdTitle"
23              android:layout_width="wrap_content"
24              android:layout_height="wrap_content"
25              android:layout_marginTop="16dp"
26              android:textColor="@color/text_primary"
27
    android:textAppearance="@style/TextAppearance.AppCompat.Headline"
28              app:layout_constraintStart_toStartOf="parent"
29              app:layout_constraintTop_toBottomOf="@id/ootdImage"
30  />

31          <TextView
32              android:id="@+id/ootdGender"
33              android:layout_width="wrap_content"
34              android:layout_height="wrap_content"
35              android:layout_marginStart="8dp"
36              android:textColor="@color/text_secondary"
37
    android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
38
    app:layout_constraintBaseline_toBaselineOf="@id/ootdTitle"
39              app:layout_constraintStart_toEndOf="@id/ootdTitle" />
40
41          <TextView
                android:id="@+id/ootdStyle"
42              android:layout_width="wrap_content"
43              android:layout_height="wrap_content"
44              android:layout_marginTop="8dp"
45              android:textColor="@color/text_secondary"
46
47  android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
48              app:layout_constraintStart_toStartOf="@id/ootdTitle"
49              app:layout_constraintTop_toBottomOf="@id/ootdTitle"
    />
50
51          <TextView
52              android:id="@+id/ootdDescription"
53              android:layout_width="match_parent"
54              android:layout_height="wrap_content"
55              android:layout_marginTop="16dp"
56              android:textColor="@color/text_secondary"
57
```

```
android:textAppearance="@style/TextAppearance.AppCompat.Body1"
            app:layout_constraintTop_toBottomOf="@id/ootdStyle"
/>

        <Button
            android:id="@+id/btnShop"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:text="@string/shop_style_button"
            style="@style/Widget.OOTDList.Button"

app:layout_constraintTop_toBottomOf="@id/ootdDescription" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
```

8. Fragment_list.xml

Tabel 12 Source Code Jawaban Soal 1

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

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:clipToPadding="false"
        android:padding="8dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

android:contentDescription="@string/recycler_content_description"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

```

```
22      <TextView
23          android:id="@+id/emptyView"
24          android:layout_width="match_parent"
25          android:layout_height="500dp"
26          android:gravity="center"
27          android:text="@string/empty_list_message"
28          android:visibility="gone"
29          app:layout_constraintBottom_toBottomOf="parent"
30          app:layout_constraintEnd_toEndOf="parent"
31          app:layout_constraintStart_toStartOf="parent"
32          app:layout_constraintTop_toTopOf="parent" />
33  </androidx.constraintlayout.widget.ConstraintLayout>
```

9. item_ootd.xml

Tabel 13 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:layout_width="match_parent"
6       android:layout_height="wrap_content"
7       android:layout_margin="8dp"
8       style="@style/CardView.OOTD">
9
10      <androidx.constraintlayout.widget.ConstraintLayout
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:padding="8dp">
14
15          <ImageView
16              android:id="@+id/ootdImage"
17              android:layout_width="match_parent"
18              android:layout_height="150dp"
19              android:scaleType="centerCrop"
20              android:background="@color/gray_300"

21  android:contentDescription="@string/ootd_image_content_description"
                app:layout_constraintTop_toTopOf="parent" />
22
23          <TextView
```

```
24              android:id="@+id/ootdTitle"
25              android:layout_width="wrap_content"
26              android:layout_height="wrap_content"
27              android:layout_marginTop="8dp"
28              android:textColor="@color/text_primary"

29  android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/ootdImage" />
30
31          <TextView
32              android:id="@+id/ootdGender"
33              android:layout_width="wrap_content"
34              android:layout_height="wrap_content"
35              android:layout_marginStart="8dp"
36              android:textColor="@color/text_secondary"

37  android:textAppearance="@style/TextAppearance.AppCompat.Caption"

38  app:layout_constraintBaseline_toBaselineOf="@id/ootdTitle"
39              app:layout_constraintStart_toEndOf="@id/ootdTitle" />

40          <TextView
41              android:id="@+id/ootdStyle"
42              android:layout_width="wrap_content"
43              android:layout_height="wrap_content"
44              android:layout_marginTop="4dp"
45              android:textColor="@color/text_secondary"

46  android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                app:layout_constraintStart_toStartOf="@id/ootdTitle"
                app:layout_constraintTop_toBottomOf="@id/ootdTitle" />
47
48          <TextView
49              android:id="@+id/ootdDescription"
50              android:layout_width="match_parent"
51              android:layout_height="wrap_content"
52              android:layout_marginTop="4dp"
53              android:textColor="@color/text_secondary"
54              android:maxLines="2"
55              android:ellipsize="end"

56  android:textAppearance="@style/TextAppearance.AppCompat.Body2"
                app:layout_constraintTop_toBottomOf="@id/ootdStyle" />
57
58          <LinearLayout
```

```
59              android:layout_width="match_parent"
60              android:layout_height="wrap_content"
61              android:layout_marginTop="8dp"
62              android:orientation="horizontal"

63  app:layout_constraintTop_toBottomOf="@id/ootdDescription">

64              <Button
65                  android:id="@+id/btnImdb"
66                  android:layout_width="0dp"
67                  android:layout_height="wrap_content"
68                  android:layout_weight="1"
69                  android:text="@string/shop_button"
70                  android:layout_marginEnd="4dp"
71                  style="@style/Widget.OOTDList.Button" />
72
73              <Button
74                  android:id="@+id/btnDetail"
75                  android:layout_width="0dp"
76                  android:layout_height="wrap_content"
77                  android:layout_weight="1"
78                  android:text="@string/detail_button"
79                  android:layout_marginStart="4dp"
80                  style="@style/Widget.OOTDList.Button" />
81          </LinearLayout>
82      </androidx.constraintlayout.widget.ConstraintLayout>
83  </androidx.cardview.widget.CardView>
```

10. nav_graph.xml

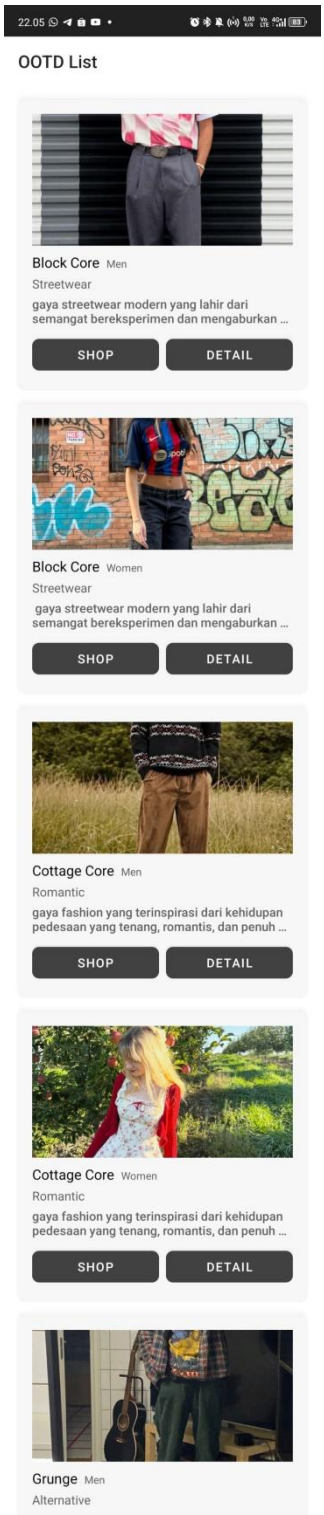
Tabel 14 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/nav_graph"
6       app:startDestination="@id/listFragment">
7
8       <fragment
9           android:id="@+id/listFragment"

10  android:name="com.example.ootdlist.fragments.ListFragment"
```

```
        android:label="OOTD List"
        tools:layout="@layout/fragment_list">
11
12          <action
13
14  android:id="@+id/action_listFragment_to_detailFragment"
15              app:destination="@id/detailFragment" />
16      </fragment>
17
18      <fragment
19          android:id="@+id/detailFragment"
20
21  android:name="com.example.ootdlist.fragments.DetailFragment"
22          tools:layout="@layout/fragment_detail">
23          <argument
24              android:name="ootdId"
25              app:argType="integer" />
26      </fragment>
27  </navigation>
```

B. Output Program



Gambar 11 Hasil Tampilan UI List Soal 1

Block Core Men

Streetwear

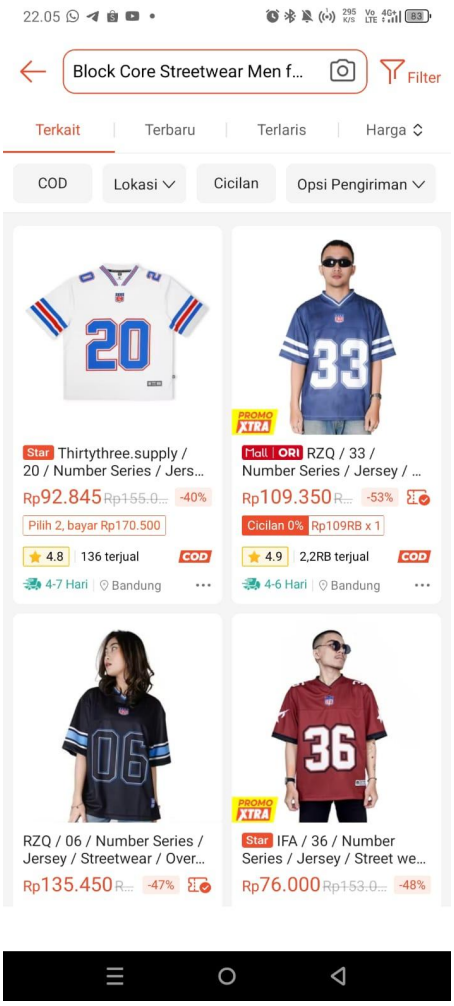
gaya streetwear modern yang lahir dari semangat bereksperimen dan mengaburkan batas antara dunia olahraga dan fashion jalanan. Keunikannya terletak pada penggunaan jersey bola sebagai item utama, menjadikannya lebih dari sekadar simbol fandom —melainkan pernyataan gaya yang berani. Gaya ini merepresentasikan generasi muda yang dinamis, penuh karakter, dan tidak takut memadukan elemen yang tampak kontras. Block Core bukan hanya soal pakaian, tapi juga tentang cara mengekspresikan diri dengan bebas dan orisinal.

Atasan: Jersey klub bola bergaya retro dengan motif kotak merah-putih dan logo "PACHA Ibiza" yang menonjol.

Bawahan: Celana wide leg bahan wol berwarna abu-abu tua, menciptakan kontras menarik antara sporty dan formal.

SHOP THIS STYLE

Gambar 12 Hasil Tampilan UI Detail Soal 1



Gambar 13 Screenshot E-commerce setelah Intent Soal 1

C. Pembahasan

1. MainActivity.kt

File ini merupakan titik masuk utama aplikasi Android. Di dalamnya, kelas MainActivity mewarisi AppCompatActivity, yang merupakan salah satu komponen utama dalam Android untuk mendukung kompatibilitas UI lintas versi. Di dalam metode onCreate(), dipanggil setContentView(R.layout.activity_main), yang berarti aplikasi akan menampilkan tampilan (UI) dari activity_main.xml sebagai antarmuka awal.

File ini secara struktur sangat sederhana, namun sangat penting karena menginisialisasi layout utama dan menjadi wadah fragment yang akan dimuat, seperti ListFragment dan DetailFragment. Biasanya MainActivity juga menjadi pengatur navigasi fragment menggunakan NavHostFragment jika menggunakan Jetpack Navigation, meskipun tidak eksplisit terlihat di file ini.

2. DetailFragment.kt

File ini adalah fragment yang bertanggung jawab menampilkan detail dari satu item OOTD (Outfit of The Day). DetailFragment menggunakan *View Binding* melalui FragmentDetailBinding untuk mengakses elemen UI secara langsung tanpa perlu findViewById, yang meningkatkan keamanan dan efisiensi kode.

Di dalam metode onViewCreated, fragment menerima argument ootdId yang dikirim dari fragment sebelumnya (kemungkinan ListFragment). ID ini digunakan untuk mengambil objek Ootd tertentu melalui metode Ootd.getById(ootdId). Setelah data OOTD diperoleh, informasi seperti judul, jenis kelamin, gaya, deskripsi, dan gambar akan ditampilkan melalui komponen UI yang telah di-*bind*.

Menariknya, terdapat fitur interaktif berupa tombol btnShop, yang ketika ditekan akan membuat URL pencarian berdasarkan properti OOTD (judul, gaya, dan gender) ke situs Shopee. Proses ini dilakukan dengan menyusun searchQuery, kemudian mengenkripsi query-nya menggunakan URLEncoder, lalu membuat *Intent* untuk membuka browser ke URL pencarian tersebut. Ini adalah contoh integrasi sederhana antara aplikasi dan layanan eksternal.

Metode onDestroyView() digunakan untuk menghapus referensi binding agar tidak terjadi memory leak, mengikuti praktik standar pengelolaan memori dalam penggunaan View Binding pada fragment.

3. ListFragment.kt:

ListFragment adalah fragment yang menampilkan daftar semua OOTD yang tersedia dalam bentuk RecyclerView. Sama seperti DetailFragment, fragment ini menggunakan FragmentListBinding untuk efisiensi akses ke komponen UI.

Pada onViewCreated, diinisialisasi sebuah OotdAdapter yang menerima data dari Ootd.ootdList, yaitu daftar statis atau dinamis dari semua outfit. Adapter ini menggunakan pola listener interface (OnItemClickListener) yang memiliki dua aksi utama: onDetailClick() dan onShopClick(). Pada onDetailClick, fragment akan melakukan navigasi ke DetailFragment dengan membawa ootdId sebagai argumen menggunakan Jetpack Navigation. Sedangkan pada onShopClick, konsepnya sama seperti pada DetailFragment, yaitu membuat query pencarian berdasarkan detail OOTD dan membuka Shopee melalui Intent ke browser.

RecyclerView dikonfigurasi dengan LinearLayoutManager untuk menampilkan daftar secara vertikal. Terdapat juga logika untuk menangani keadaan jika daftar kosong (ootdList.isEmpty()), yang akan menyembunyikan RecyclerView dan menampilkan emptyView, semacam teks atau ikon placeholder yang menunjukkan bahwa tidak ada data tersedia.

Seperti sebelumnya, metode onDestroyView() digunakan untuk mencegah memory leak dengan menghapus binding saat fragment dihancurkan.

4. OotdAdapter.kt

File OotdAdapter.kt merupakan salah satu komponen utama dalam arsitektur RecyclerView Adapter yang bertugas menghubungkan data outfit (OOTD) dengan tampilan daftar item di UI. Kelas OotdAdapter menerima dua parameter utama: ootdList, yaitu daftar objek Ootd, dan listener, yang merupakan antarmuka OnItemClickListener untuk menangani aksi klik dari pengguna. Adapter ini mewarisi RecyclerView.Adapter dan di dalamnya didefinisikan kelas OotdViewHolder yang bertugas untuk mem-bind setiap elemen data ke tampilan item yang sudah ditentukan dalam item_ootd.xml.

Pada metode bind(), setiap properti dari objek Ootd seperti title, gender, style, description, dan imageResId ditampilkan ke dalam elemen visual seperti TextView dan ImageView. Tombol btnDetail dan btnImdb (yang seharusnya secara konteks bisa diubah menjadi btnShop) masing-masing memiliki click listener yang memanggil fungsi dari OnItemClickListener. Ini menunjukkan bahwa adapter ini juga menangani interaksi pengguna, seperti ketika ingin melihat detail atau menuju toko. Selain itu, adapter juga menerapkan konten deskriptif gambar (contentDescription) demi mendukung aksesibilitas.

Metode onCreateViewHolder() berfungsi untuk meng-inflate layout item dari file XML dan mengembalikan OotdViewHolder. Sedangkan onBindViewHolder() memanggil fungsi bind() untuk memasukkan data OOTD berdasarkan posisi. Metode getItemCount() cukup sederhana, mengembalikan jumlah total data yang akan ditampilkan.

5. **OotdData.kt**

File OotdData.kt berisi deklarasi data model Ootd dan data source statis yang mewakili berbagai gaya fashion. Kelas Ootd menggunakan anotasi data class yang secara otomatis menyediakan fungsi seperti toString(), equals(), dan copy(). Setiap objek Ootd memiliki properti: id, title, style, description, imageResId, dan gender. Informasi ini mewakili detail outfit termasuk judul tampilan, kategori gaya (misalnya Streetwear, Romantic, Alternative), deskripsi naratif yang mendalam, serta referensi ke resource gambar.

Objek-objek Ootd dikelompokkan ke dalam daftar statis ootdList yang berfungsi sebagai data dummy atau konten statis dalam aplikasi. Masing-masing entri menggambarkan gaya fashion spesifik, lengkap dengan narasi inspiratif serta penjelasan atasan dan bawahan, yang sangat berguna untuk menciptakan pengalaman pengguna yang imersif dan edukatif. Ada berbagai gaya seperti Block Core, Cottage Core, Grunge, Japanese Americana, Old Money, dan Y2K, yang semuanya menjelaskan elemen fashion yang populer pada segmennya.

Fungsi getById() disediakan untuk mencari OOTD berdasarkan ID-nya, sangat berguna ketika aplikasi perlu menampilkan detail item tertentu saat pengguna memilih dari daftar.

6. activity_main.xml

File activity_main.xml merupakan file tata letak utama (main layout) dari aplikasi yang mendefinisikan struktur dasar UI dengan menggunakan FrameLayout. Di dalamnya, terdapat komponen FragmentContainerView

yang berfungsi sebagai host dari navigasi berbasis fragment. View ini menggunakan NavHostFragment dari Jetpack Navigation Component untuk mengelola transisi antar fragment berdasarkan navigation graph (nav_graph.xml).

Atribut app:defaultNavHost="true" memastikan bahwa fragment ini akan menjadi host utama untuk navigasi (misalnya saat menekan tombol back). Atribut app:navGraph="@navigation/nav_graph" menunjukkan file navigasi XML yang berisi arah dan struktur perpindahan fragment. Tata letak ini juga memiliki latar belakang yang ditentukan melalui warna @color/background, memberikan kesan visual yang konsisten di seluruh aplikasi.

7. fragment_detail.xml

File fragment_detail.xml adalah layout tampilan detail untuk setiap OOTD yang dipilih. Layout dibungkus dalam ScrollView, memungkinkan pengguna menggulir jika kontennya panjang. Di dalamnya terdapat ConstraintLayout yang mengatur elemen-elemen UI secara fleksibel dan responsif. Elemen-elemen yang ditampilkan mencakup ImageView (ootdImage) untuk gambar outfit, serta TextView untuk ootdTitle, ootdGender, ootdStyle, dan ootdDescription. Semua teks menggunakan TextAppearance untuk menjaga konsistensi desain dan tema aplikasi.

Di bagian akhir terdapat tombol btnShop, yang memungkinkan pengguna melakukan aksi lanjut seperti membuka halaman belanja atau eksternal link ke produk serupa. Tata letak ini memberikan pengalaman detail yang informatif dan estetis bagi pengguna untuk memahami outfit secara menyeluruh. Gaya desainnya bersih, rapi, dan memprioritaskan keterbacaan serta hirarki informasi.

8. fragment_list.xml

File fragment_list.xml merupakan layout XML untuk tampilan utama dari ListFragment, yaitu sebuah fragment yang menampilkan daftar item OOTD (Outfit of The Day) dalam aplikasi. Layout ini menggunakan ConstraintLayout sebagai root layout, yang memungkinkan setiap elemen UI diposisikan dengan fleksibel dan efisien berdasarkan constraint antar elemen. Di dalamnya terdapat dua elemen utama: RecyclerView dan TextView.

RecyclerView dengan ID @+id/recyclerView berfungsi untuk menampilkan daftar OOTD secara vertikal menggunakan LinearLayoutManager. Atribut clipToPadding="false" dan padding sebesar 8dp memastikan tampilan daftar lebih lega dan tidak terpotong. RecyclerView ini akan diisi oleh adapter yang mengatur item-item OOTD. Sebagai fallback, TextView dengan ID @+id/emptyView disediakan dan akan ditampilkan ketika daftar kosong. TextView ini menampilkan pesan dari string resource @string/empty_list_message, dan secara default disembunyikan (android:visibility="gone"), hanya muncul saat diperlukan. Layout ini sangat mendukung prinsip *responsive UI*, karena semua elemen menggunakan constraint terhadap parent, memastikan kompatibilitas pada berbagai ukuran layar.

9. item_ootd.xml

File ini adalah layout dari setiap item OOTD yang akan ditampilkan dalam RecyclerView. Root layout-nya menggunakan CardView, yang memberikan efek bayangan dan sudut membulat, menciptakan tampilan yang

lebih menarik dan modern. Selanjutnya, ConstraintLayout digunakan untuk mengatur tata letak elemen-elemen di dalam card tersebut.

Elemen utama dari layout ini adalah ImageView (ID: ootdImage) yang menampilkan gambar OOTD dengan tinggi tetap 150dp dan properti centerCrop agar gambar proporsional memenuhi area tampilan. Kemudian terdapat beberapa TextView, masing-masing untuk menampilkan judul outfit (ootdTitle), jenis kelamin target outfit (ootdGender), gaya berpakaian (ootdStyle), dan deskripsi singkat (ootdDescription). Deskripsi dibatasi maksimal dua baris dengan ellipsize="end" untuk menjaga tampilan tetap rapi.

Pada bagian bawah terdapat LinearLayout horizontal yang memuat dua tombol (Button). Tombol pertama (btnImdb) biasanya digunakan untuk mengarahkan ke halaman toko atau e-commerce, dan tombol kedua (btnDetail) digunakan untuk membuka detail lengkap dari OOTD yang dipilih. Masing-masing tombol memiliki gaya yang konsisten sesuai style Widget.OOTDList.Button. Dengan struktur ini, tiap item OOTD tidak hanya informatif secara visual, tetapi juga interaktif, memungkinkan pengguna mengambil tindakan langsung.

10. nav_graph.xml

File nav_graph.xml adalah bagian dari Android Jetpack Navigation Component yang mendefinisikan struktur navigasi antar fragment dalam aplikasi. Di dalam file ini terdapat dua fragment utama: listFragment dan detailFragment. listFragment berfungsi sebagai *startDestination*, artinya ketika aplikasi dijalankan, pengguna akan langsung diarahkan ke fragment ini yang menampilkan daftar OOTD.

Dari listFragment, terdapat satu action menuju detailFragment, yaitu action_listFragment_to_detailFragment. Action ini memungkinkan pengguna berpindah ke tampilan detail dari item yang dipilih. Sementara itu, detailFragment didefinisikan dengan satu argument bernama ootdId bertipe integer, yang akan digunakan untuk mengambil dan menampilkan data detail berdasarkan ID yang dikirim dari fragment sebelumnya. Konfigurasi ini memungkinkan navigasi antar UI dilakukan dengan mudah dan aman, karena argument telah diketik secara eksplisit (type-safe).

SOAL 2

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

A. Jawaban

RecyclerView telah ada sejak lama dan sangat stabil serta kompatibel dengan hampir semua versi android (sejak API 14). Dimana banyak proyek besar masih memakai XML + View System sehingga membuat recyclerview masih relevan untuk pemeliharaan dan pengembangan lanjutan. Lalu RecyclerView didukung oleh banyak library, plugin, dan ekstensi (seperti DiffUtil, ListAdapter, Paging3, dll) yang memperkaya fitur dan performa. Sedangkan LazyColumn dari Jetpack Compose masih berkembang dan belum selalu cocok untuk semua skenario kompleks. Lalu RecyclerView sangat fleksibel untuk berbagai jenis layout dan interaksi, seperti nested scrolling, custom item decorations, swipe gestures, drag-and-drop, dan lain-lain. Compose memang mendukung ini, tapi pada beberapa kasus kompleks, Compose masih mengalami keterbatasan atau bug.

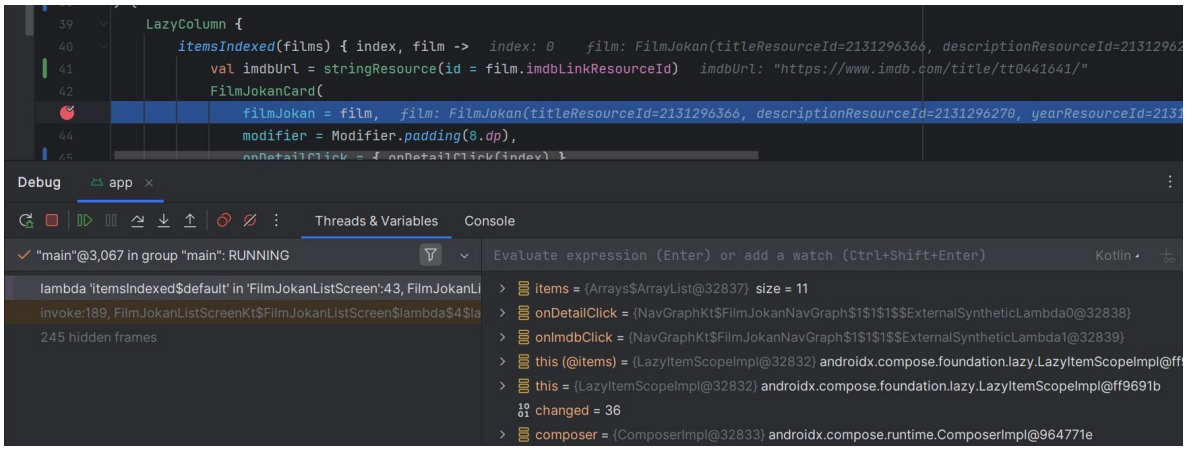
MODUL 4 : ViewModel and Debugging

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

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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



Gambar 14 Contoh Penggunaan Debugger

A. Source Code

1. MainActivity.kt

Tabel 15 Source Code Jawaban Soal 1

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

2. DetailFragment.kt

Tabel 16 Source Code Jawaban Soal 1

```
1   package com.example.ootdlist.fragments
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import androidx.fragment.app.Fragment
10  import com.example.ootdlist.data.Ootd
11  import com.example.ootdlist.databinding.FragmentDetailBinding
12  import java.net.URLEncoder
13
```

```
14  class DetailFragment : Fragment() {
15      private var _binding: FragmentDetailBinding? = null
16      private val binding get() = _binding!!
17
18      override fun onCreateView(
19          inflater: LayoutInflater,
20          container: ViewGroup?,
21          savedInstanceState: Bundle?
22      ): View {
23          _binding = FragmentDetailBinding.inflate(inflater,
    container, false)
24          return binding.root
25      }
26
27      override fun onViewCreated(view: View, savedInstanceState:
    Bundle?) {
28          super.onViewCreated(view, savedInstanceState)
29
30          val ootdId = requireArguments().getInt("ootdId")
31          val ootd = Ootd.getById(ootdId)
32
33          if (ootd != null) {
34              binding.ootdTitle.text = ootd.title
35              binding.ootdGender.text = ootd.gender
36              binding.ootdStyle.text = ootd.style
37              binding.ootdDescription.text = ootd.description
38              binding.ootdImage.setImageResource(ootd.imageResId)
39
40              binding.btnShop.setOnClickListener {
41                  val searchQuery = "${ootd.title} ${ootd.style}
    ${ootd.gender} fashion"
42                  val encodedQuery =
    URLEncoder.encode(searchQuery, "UTF-8")
43                  val url =
    "https://shopee.co.id/search?keyword=$encodedQuery"
44                  val intent = Intent(Intent.ACTION_VIEW,
    Uri.parse(url))
45                  startActivity(intent)
46              }
47          }
48      }
49
50      override fun onDestroyView() {
51          super.onDestroyView()
52          _binding = null
53      }
54  }
55
```

3. ListFragment.kt

Tabel 17 Source Code Jawaban Soal 1

```
1  package com.example.ootdlist.fragments
2
3  import android.content.Intent
4  import android.net.Uri
5  import android.os.Bundle
6  import android.view.LayoutInflater
7  import android.view.View
8  import android.view.ViewGroup
9  import androidx.fragment.app.Fragment
10 import androidx.navigation.fragment.findNavController
11 import androidx.recyclerview.widget.LinearLayoutManager
12 import com.example.ootdlist.R
13 import com.example.ootdlist.adapter.OotdAdapter
14 import com.example.ootdlist.data.Ootd
15 import com.example.ootdlist.databinding.FragmentListBinding
16 import java.net.URLEncoder
17
18 class ListFragment : Fragment() {
19     private var _binding: FragmentListBinding? = null
20     private val binding get() = _binding!!
21
22     override fun onCreateView(
23         inflater: LayoutInflater,
24         container: ViewGroup?,
25         savedInstanceState: Bundle?
26     ): View {
27         _binding = FragmentListBinding.inflate(inflater,
    container, false)
28         return binding.root
29     }
30
31     override fun onViewCreated(view: View, savedInstanceState:
32 Bundle?) {
33         super.onViewCreated(view, savedInstanceState)
34
35         val ootdList = Ootd.ootdList
36         val adapter = OotdAdapter(ootdList, object :
    OotdAdapter.OnItemClickListener {
37             override fun onDetailClick(ootd: Ootd) {
38                 val bundle = Bundle().apply {
39                     putInt("ootdId", ootd.id)
40                 }
41
    findNavController().navigate(R.id.action_listFragment_to_detailFr
    agment, bundle)
42             }
```

```
43
44              override fun onShopClick(ootd: Ootd) {
45                  val searchQuery = "${ootd.title} ${ootd.style}
46  ${ootd.gender} fashion"
                    val encodedQuery = URLEncoder.encode(searchQuery,
47  "UTF-8")
                    val url =
48  "https://shopee.co.id/search?keyword=$encodedQuery"
                    val intent = Intent(Intent.ACTION_VIEW,
49  Uri.parse(url))
50                  startActivity(intent)
51              }
52          })
53
54          binding.recyclerView.layoutManager =
    LinearLayoutManager(context)
55          binding.recyclerView.adapter = adapter
56
57          if (ootdList.isEmpty()) {
58              binding.recyclerView.visibility = View.GONE
59              binding.emptyView.visibility = View.VISIBLE
60          } else {
61              binding.recyclerView.visibility = View.VISIBLE
62              binding.emptyView.visibility = View.GONE
63          }
64      }
65
66      override fun onDestroyView() {
67          super.onDestroyView()
68          _binding = null
69      }
    }
```

4. OotdAdapter.kt

Tabel 18 Source Code Jawaban Soal 1

```
1   package com.example.ootdlist.adapter
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.RecyclerView
6   import com.example.ootdlist.R
7   import com.example.ootdlist.data.Ootd
8   import com.example.ootdlist.databinding.ItemOotdBinding
9
10  class OotdAdapter(private val ootdList: List<Ootd>, private val
```

```
    listener: OnItemClickListener) :
11      RecyclerView.Adapter<OotdAdapter.OotdViewHolder>() {
12
13      interface OnItemClickListener {
14          fun onDetailClick(ootd: Ootd)
15          fun onShopClick(ootd: Ootd)
16      }
17
18      inner class OotdViewHolder(private val binding:
19  ItemOotdBinding) : RecyclerView.ViewHolder(binding.root) {
            fun bind(ootd: Ootd) {
20              binding.ootdTitle.text = ootd.title
21              binding.ootdGender.text = ootd.gender
22              binding.ootdStyle.text = ootd.style
23              binding.ootdDescription.text = ootd.description
24              binding.ootdImage.setImageResource(ootd.imageResId)
25
26              binding.btnImdb.setOnClickListener {
27                  listener.onShopClick(ootd)
28              }
29
30              binding.btnDetail.setOnClickListener {
31                  listener.onDetailClick(ootd)
32              }
33
34              val imageContentDesc = itemView.context.getString(
35                  R.string.ootd_image_content_description,
36                  ootd.title,
37                  ootd.gender
38              )
39              binding.ootdImage.contentDescription =
40  imageContentDesc
            }
41      }
42
43      override fun onCreateViewHolder(parent: ViewGroup, viewType:
44  Int): OotdViewHolder {
            val binding =
45  ItemOotdBinding.inflate(LayoutInflater.from(parent.context),
    parent, false)
            return OotdViewHolder(binding)
46      }
47
48      override fun onBindViewHolder(holder: OotdViewHolder,
49  position: Int) {
            holder.bind(ootdList[position])
50      }
51
52      override fun getItemCount(): Int = ootdList.size
```

```
53  }
```

5. OotdData.kt

Tabel 19 Source Code Jawaban Soal 1

```
1   package com.example.ootdlist.data
2
3   import com.example.ootdlist.R
4
5   data class Ootd(
6       val id: Int,
7       val title: String,
8       val style: String,
9       val description: String,
10      val imageResId: Int,
11      val gender: String
12  ) {
13      companion object {
14          val ootdList = listOf(
15              Ootd(1, "Block Core", "Streetwear", "gaya streetwear
    modern yang lahir dari semangat bereksperimen dan mengaburkan
    batas antara dunia olahraga dan fashion jalanan. " +
16                      "Keunikannya terletak pada penggunaan jersey
    bola sebagai item utama, menjadikannya lebih dari sekadar simbol
    fandom—melainkan pernyataan gaya yang berani. " +
17                      "Gaya ini merepresentasikan generasi muda
    yang dinamis, penuh karakter, dan tidak takut memadukan elemen
    yang tampak kontras. Block Core bukan hanya soal pakaian," +
18                      " tapi juga tentang cara mengekspresikan
    diri dengan bebas dan orisinal." +
19                      "\n\n Atasan: Jersey klub bola bergaya retro
    dengan motif kotak merah-putih dan logo “PACHA Ibiza” yang
    menonjol." +
20                      "\n\n Bawahan: Celana wide leg bahan wol
    berwarna abu-abu tua, menciptakan kontras menarik antara sporty
    dan formal.", R.drawable.block_core_man, "Men"),
21
22              Ootd(2, "Block Core", "Streetwear", " gaya
    streetwear modern yang lahir dari semangat bereksperimen dan
    mengaburkan batas antara dunia olahraga dan fashion jalanan. " +
23                      "Keunikannya terletak pada penggunaan jersey
    bola sebagai item utama, menjadikannya lebih dari sekadar simbol
    fandom—melainkan pernyataan gaya yang berani. " +
24                      "Gaya ini merepresentasikan generasi muda
    yang dinamis, penuh karakter, dan tidak takut memadukan elemen
    yang tampak kontras. Block Core bukan hanya soal pakaian, " +
```

```
25                      "tapi juga tentang cara mengekspresikan diri
    dengan bebas dan orisinal." +
26                      "\n\n Atasan: Jersey FC Barcelona dengan
    potongan crop top, menampilkan logo Nike dan sponsor Spotify
    yang mencolok." +
27                      "\n\n Bawahan: Celana cargo hitam longgar,
    menciptakan siluet santai namun tetap bold.",
    R.drawable.block_core_girl, "Women"),
28
29              Ootd(3, "Cottage Core", "Romantic", "gaya fashion
    yang terinspirasi dari kehidupan pedesaan yang tenang, romantis,
    dan penuh nostalgia. " +
30                      "Gaya ini mengangkat elemen alami dan
    vintage—mulai dari motif bunga kecil, bahan katun atau linen,
    warna-warna hangat dan lembut," +
31                      " hingga siluet yang mengalir dan nyaman.
    Cottage Core bukan hanya tentang penampilan, tetapi juga
    menggambarkan gaya hidup yang menghargai" +
32                      " keindahan sederhana, kedekatan dengan
    alam, dan suasana yang damai." +
33                      "\n\n Atasan: Sweater rajut motif folk
    berwarna dasar hitam dengan aksen pola merah-putih yang memberi
    kesan hangat." +
34                      "\n\n Bawahan: Celana panjang corduroy
    coklat muda berpotongan longgar, menambah kesan earthy dan
    santai.", R.drawable.cottage_core_man, "Men"),
35
36              Ootd(4, "Cottage Core", "Romantic", "gaya fashion
    yang terinspirasi dari kehidupan pedesaan yang tenang, romantis,
    dan penuh nostalgia." +
37                      " Gaya ini mengangkat elemen alami dan
    vintage—mulai dari motif bunga kecil, bahan katun atau linen,
    warna-warna hangat dan lembut," +
38                      " hingga siluet yang mengalir dan nyaman.
    Cottage Core bukan hanya tentang penampilan, tetapi juga
    menggambarkan gaya hidup yang menghargai " +
39                      "keindahan sederhana, kedekatan dengan alam,
    dan suasana yang damai." +
40                      "\n\n Atasan: Kardigan rajut tipis berwarna
    merah dipadukan dengan dress floral putih bermotif bunga kecil
    dan detail renda." +
41                      "\n\n Bawahan: Dress panjang mengalir dengan
    potongan klasik dan aksen pita, menciptakan siluet lembut yang
    feminin.", R.drawable.cottage_core_girl, "Women"),
42
43              Ootd(5, "Grunge", "Alternative", "Outfit bergaya
    grunge identik dengan kesan santai, berantakan yang terkesan
    effortless namun tetap keren. " +
44                      "Ciri khasnya adalah dominasi warna gelap,
```

```
        layering, bahan denim, serta kaos band sebagai elemen penting. "
        +
45                      "Gaya ini memadukan unsur musik alternatif
        dengan sikap anti-mainstream khas era 90-an." +
46                      "\n\n Atasan: Kaos grafis berwarna hitam
        dilapisi kemeja flanel motif kotak merah-hijau yang terbuka,
        menciptakan tampilan layering khas grunge klasik." +
47                      "\n\n Bawahan: Celana longgar berbahan
        corduroy hijau tua, memberi tekstur earthy yang sejalan dengan
        elemen grunge vintage.", R.drawable.grunge_man, "Men"),
48
49              Ootd(6, "Grunge", "Alternative", "Outfit bergaya
        grunge identik dengan kesan santai, berantakan yang terkesan
        effortless namun tetap keren. " +
50                      "Ciri khasnya adalah dominasi warna gelap,
        layering, bahan denim, serta kaos band sebagai elemen penting."
        +
51                      " Gaya ini memadukan unsur musik alternatif
        dengan sikap anti-mainstream khas era 90-an." +
52                      "\n\n Atasan: Kaos band The Beatles berwarna
        hitam kusam dengan print besar, memberikan nuansa retro dan
        rebellious." +
53                      "\n\n Bawahan: Celana jeans baggy
        berpotongan longgar berwarna biru pudar, menambah kesan santai
        dan nonchalant.", R.drawable.grunge_girl, "Women"),
54
55              Ootd(7, "Japanese Americana", "Fusion", "Outfit
        bergaya Japanese Americana memadukan gaya kasual klasik khas
        Amerika seperti denim dan workwear dengan sentuhan rapi,
        minimalis, dan estetika layering ala Jepang. " +
56                      "Ciri khasnya meliputi jaket denim, celana
        berpotongan longgar, warna earthy seperti olive dan navy, serta
        perhatian terhadap siluet dan tekstur." +
57                      "\n\n Atasan: Kemeja putih berkerah dengan
        dasi cokelat yang dilapisi jaket denim fitted berwarna indigo,
        menciptakan tampilan rapi dengan lapisan Americana yang kuat." +
58                      "\n\n Bawahan: Celana workwear hijau zaitun
        berpotongan lurus yang terinspirasi dari gaya militer,
        menyeimbangkan kesan maskulin dan vintage.",
        R.drawable.japanese_americana_man, "Men"),
59
60              Ootd(8, "Japanese Americana", "Fusion", "Outfit
        bergaya Japanese Americana memadukan gaya kasual klasik khas
        Amerika seperti denim dan workwear dengan sentuhan rapi,
        minimalis, dan estetika layering ala Jepang. " +
61                      "Ciri khasnya meliputi jaket denim, celana
        berpotongan longgar, warna earthy seperti olive dan navy, serta
        perhatian terhadap siluet dan tekstur." +
62                      "\n\n Atasan: Kaos bergaris hitam-putih
```

```
        bergaya Breton yang dimasukkan ke dalam celana, memberikan kesan
        clean namun santai. Sebuah jaket denim biru dikenakan secara
        kasual di bahu sebagai outer, menambah sentuhan Americana
        klasik." +
63                      "\n\n Bawahan: Celana cargo hijau army
        berpotongan lebar yang memberi nuansa utilitarian khas workwear,
        sekaligus menciptakan siluet longgar yang modis dan nyaman.",
        R.drawable.japanese_americana_girl, "Women"),
64
65              Ootd(9, "Old Money", "Classic", "Outfit bergaya Old
        Money mencerminkan kemewahan yang tidak mencolok, elegan, dan
        timeless. Gaya ini lekat dengan nuansa klasik, potongan yang
        rapi," +
66                      " bahan berkualitas tinggi, serta warna-
        warna netral yang terkesan mahal. Tidak berfokus pada tren,
        tetapi pada keanggunan dan prestise." +
67                      "\n\n Atasan: Kemeja putih klasik dengan
        kerah terbuka, memberikan kesan effortless dan bersih namun
        tetap terjaga rapi. Diselipkan ke dalam celana untuk tampilan
        yang lebih polished." +
68                      "\n\n Bawahan: Celana wool cokelat
        berpotongan longgar dan pleated, menambah kesan sophisticated
        khas gentleman lama. ", R.drawable.old_money_man, "Men"),
69
70              Ootd(10, "Old Money", "Classic", "Outfit bergaya Old
        Money mencerminkan kemewahan yang tidak mencolok, elegan, dan
        timeless. Gaya ini lekat dengan nuansa klasik, potongan yang
        rapi," +
71                      " bahan berkualitas tinggi, serta warna-
        warna netral yang terkesan mahal. Tidak berfokus pada tren,
        tetapi pada keanggunan dan prestise." +
72                      "\n\n Atasan: Kemeja satin berwarna ivory
        dengan detail kerah klasik, memberikan tampilan sophisticated
        yang mewah namun understated. Dipadukan dengan blazer oversized
        dalam nuansa beige yang memberikan siluet structured dan
        elegan." +
73                      "\n\n Bawahan: Celana tailored high-waist
        warna senada yang jatuh longgar hingga menyentuh lantai,
        menegaskan citra refined dan berkelas.",
        R.drawable.old_money_girl, "Women"),
74
75              Ootd(11, "Y2K", "Retro Futurism", "Outfit bergaya
        Y2K (Year 2000) identik dengan nuansa futuristik, playful, dan
        eksperimental yang merefleksikan semangat awal milenium. " +
76                      "Gaya ini sering menggabungkan elemen-elemen
        seperti low-rise jeans, atasan ketat, bahan denim dengan efek
        faded, " +
77                      "serta aksesori mencolok yang terinspirasi
        dari dunia pop culture, teknologi, dan fashion selebriti tahun
```

```
    2000-an." +
78                      "\n\n Atasan: Kemeja putih sebagai dasar
    dipadukan dengan sweater hitam dan jaket kulit hitam sebagai
    outer, menghasilkan tampilan layering maskulin dan berani yang
    populer di era awal 2000-an." +
79                      "\n\n Bawahan: Celana jeans longgar berwarna
    hitam pudar, menciptakan kesan laid-back namun tetap tajam,
    selaras dengan gaya street style Y2K.", R.drawable.y2k_man,
    "Men"),
80
81              Ootd(12, "Y2K", "Retro Futurism", "Outfit bergaya
    Y2K (Year 2000) identik dengan nuansa futuristik, playful, dan
    eksperimental yang merefleksikan semangat awal milenium. " +
82                      "Gaya ini sering menggabungkan elemen-elemen
    seperti low-rise jeans, atasan ketat, bahan denim dengan efek
    faded, " +
83                      "serta aksesori mencolok yang terinspirasi
    dari dunia pop culture, teknologi, dan fashion selebriti tahun
    2000-an." +
84                      "\n\n Atasan: Cropped cardigan abu-abu tua
    yang ketat dipadukan dengan tank top abu-abu muda, menciptakan
    tampilan layering yang menggoda dan edgy." +
85                      "\n\n Bawahan: Celana jeans flare
    berpotongan low-rise dengan efek washed-out, memperkuat siluet
    tubuh dan nuansa retro futuristik khas Y2K.",
    R.drawable.y2k_girl, "Women")
86          )
87
88          fun getById(id: Int): Ootd? = ootdList.find { it.id ==
    id }
89      }
90  }
```

6. OotdView Model.kt

Tabel 20 Source Code Jawaban Soal 1

```
1   package com.example.modul4
2
    import android.util.Log
3   import androidx.lifecycle.ViewModel
4   import kotlinx.coroutines.flow.MutableStateFlow
5   import kotlinx.coroutines.flow.StateFlow
6
7
8   class OotdViewModel : ViewModel() {
9
10      private val _ootdList =
    MutableStateFlow<List<Ootd>>(emptyList())
```

```
    val ootdList: StateFlow<List<Ootd>> get() = _ootdList

    private val _selectedOotd = MutableStateFlow<Ootd?>(null)
    val selectedOotd: StateFlow<Ootd?> get() = _selectedOotd

    init {
        _ootdList.value = Ootd.ootdList
        Log.d("OotdViewModel", "Data dimuat ke dalam list: ${_ootdList.value.size} item")
    }

    fun selectOotd(ootd: Ootd) {
        _selectedOotd.value = ootd
        Log.d("OotdViewModel", "Item dipilih: ${ootd.title} (${ootd.gender})")
    }
}
```

7. OotdView ModelFactory.kt

Tabel 21 Source Code Jawaban Soal 1

```
package com.example.modul4

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

class OotdViewModelFactory : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(OotdViewModel::class.java)) {
            return OotdViewModel() as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

8. activity_main.xml

Tabel 22 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph" />
</FrameLayout>
```

9. Fragment_detail.xml

Tabel 23 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    android:padding="16dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/ootdImage"
            android:layout_width="match_parent"
            android:layout_height="500dp"
            android:background="@color/gray_300"
            android:scaleType="centerCrop"
            app:layout_constraintTop_toTopOf="parent" />

```

```
21          <TextView
22              android:id="@+id/ootdTitle"
23              android:layout_width="wrap_content"
24              android:layout_height="wrap_content"
25              android:layout_marginTop="16dp"
26              android:textColor="@color/text_primary"
27
    android:textAppearance="@style/TextAppearance.AppCompat.Headline"
28              app:layout_constraintStart_toStartOf="parent"
29              app:layout_constraintTop_toBottomOf="@id/ootdImage"
30  />

31          <TextView
32              android:id="@+id/ootdGender"
33              android:layout_width="wrap_content"
34              android:layout_height="wrap_content"
35              android:layout_marginStart="8dp"
36              android:textColor="@color/text_secondary"
37
    android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
38
    app:layout_constraintBaseline_toBaselineOf="@id/ootdTitle"
39              app:layout_constraintStart_toEndOf="@id/ootdTitle" />
40
41          <TextView
                android:id="@+id/ootdStyle"
42              android:layout_width="wrap_content"
43              android:layout_height="wrap_content"
44              android:layout_marginTop="8dp"
45              android:textColor="@color/text_secondary"
46
47  android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
48              app:layout_constraintStart_toStartOf="@id/ootdTitle"
49              app:layout_constraintTop_toBottomOf="@id/ootdTitle"
    />
50
51          <TextView
52              android:id="@+id/ootdDescription"
53              android:layout_width="match_parent"
54              android:layout_height="wrap_content"
55              android:layout_marginTop="16dp"
56              android:textColor="@color/text_secondary"
57
58  android:textAppearance="@style/TextAppearance.AppCompat.Body1"
59              app:layout_constraintTop_toBottomOf="@id/ootdStyle"
60  />
61
62          <Button
63              android:id="@+id/btnShop"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:text="@string/shop_style_button"
            style="@style/Widget.OOTDList.Button"

app:layout_constraintTop_toBottomOf="@id/ootdDescription" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>
```

10. Fragment_list.xml

Tabel 24 Source Code Jawaban Soal 1

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

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:clipToPadding="false"
        android:padding="8dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

android:contentDescription="@string/recycler_content_description"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/emptyView"
        android:layout_width="match_parent"
        android:layout_height="500dp"
        android:gravity="center"
        android:text="@string/empty_list_message"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
32          app:layout_constraintTop_toTopOf="parent" />
33  </androidx.constraintlayout.widget.ConstraintLayout>
```

11. item_ootd.xml

Tabel 25 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:layout_width="match_parent"
6       android:layout_height="wrap_content"
7       android:layout_margin="8dp"
8       style="@style/CardView.OOTD">
9
10      <androidx.constraintlayout.widget.ConstraintLayout
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:padding="8dp">
14
15          <ImageView
16              android:id="@+id/ootdImage"
17              android:layout_width="match_parent"
18              android:layout_height="150dp"
19              android:scaleType="centerCrop"
20              android:background="@color/gray_300"

21  android:contentDescription="@string/ootd_image_content_description"
                app:layout_constraintTop_toTopOf="parent" />
22
23          <TextView
24              android:id="@+id/ootdTitle"
25              android:layout_width="wrap_content"
26              android:layout_height="wrap_content"
27              android:layout_marginTop="8dp"
28              android:textColor="@color/text_primary"

29  android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/ootdImage" />
30
31          <TextView
32              android:id="@+id/ootdGender"
33              android:layout_width="wrap_content"
34              android:layout_height="wrap_content"
```

```
35              android:layout_marginStart="8dp"
36              android:textColor="@color/text_secondary"

37  android:textAppearance="@style/TextAppearance.AppCompat.Caption"

38  app:layout_constraintBaseline_toBaselineOf="@id/ootdTitle"
39              app:layout_constraintStart_toEndOf="@id/ootdTitle" />

40          <TextView
41              android:id="@+id/ootdStyle"
42              android:layout_width="wrap_content"
43              android:layout_height="wrap_content"
44              android:layout_marginTop="4dp"
45              android:textColor="@color/text_secondary"

46  android:textAppearance="@style/TextAppearance.AppCompat.Body1"
                app:layout_constraintStart_toStartOf="@id/ootdTitle"
                app:layout_constraintTop_toBottomOf="@id/ootdTitle" />
47
48          <TextView
49              android:id="@+id/ootdDescription"
50              android:layout_width="match_parent"
51              android:layout_height="wrap_content"
52              android:layout_marginTop="4dp"
53              android:textColor="@color/text_secondary"
54              android:maxLines="2"
55              android:ellipsize="end"

56  android:textAppearance="@style/TextAppearance.AppCompat.Body2"
                app:layout_constraintTop_toBottomOf="@id/ootdStyle" />
57
58          <LinearLayout
59              android:layout_width="match_parent"
60              android:layout_height="wrap_content"
61              android:layout_marginTop="8dp"
62              android:orientation="horizontal"

63  app:layout_constraintTop_toBottomOf="@id/ootdDescription">

64              <Button
65                  android:id="@+id/btnImdb"
66                  android:layout_width="0dp"
67                  android:layout_height="wrap_content"
68                  android:layout_weight="1"
69                  android:text="@string/shop_button"
70                  android:layout_marginEnd="4dp"
71                  style="@style/Widget.OOTDList.Button" />
72
73              <Button
```

```
74                  android:id="@+id/btnDetail"
75                  android:layout_width="0dp"
76                  android:layout_height="wrap_content"
77                  android:layout_weight="1"
78                  android:text="@string/detail_button"
79                  android:layout_marginStart="4dp"
80                  style="@style/Widget.OOTDList.Button" />
81          </LinearLayout>
82      </androidx.constraintlayout.widget.ConstraintLayout>
83  </androidx.cardview.widget.CardView>
```

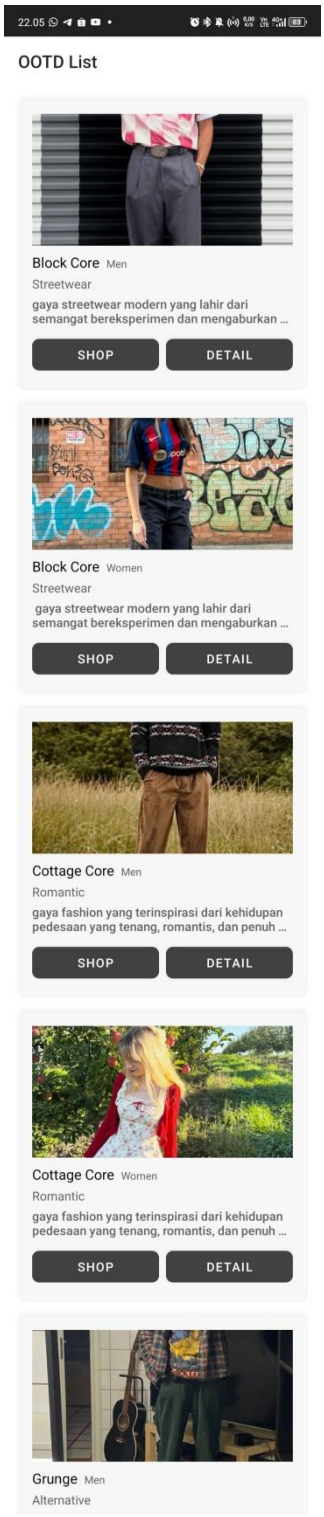
12. nav_graph.xml

Tabel 26 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/nav_graph"
6       app:startDestination="@id/listFragment">
7
8       <fragment
9           android:id="@+id/listFragment"

10  android:name="com.example.ootdlist.fragments.ListFragment"
            android:label="OOTD List"
            tools:layout="@layout/fragment_list">
11
12          <action
13
14  android:id="@+id/action_listFragment_to_detailFragment"
15              app:destination="@id/detailFragment" />
16      </fragment>
17
18      <fragment
19          android:id="@+id/detailFragment"
20
21  android:name="com.example.ootdlist.fragments.DetailFragment"
22          tools:layout="@layout/fragment_detail">
23          <argument
24              android:name="ootdId"
25              app:argType="integer" />
26      </fragment>
27  </navigation>
```

B. Output Program



Gambar 15 Hasil Tampilan UI List Soal 1

Block Core Men

Streetwear

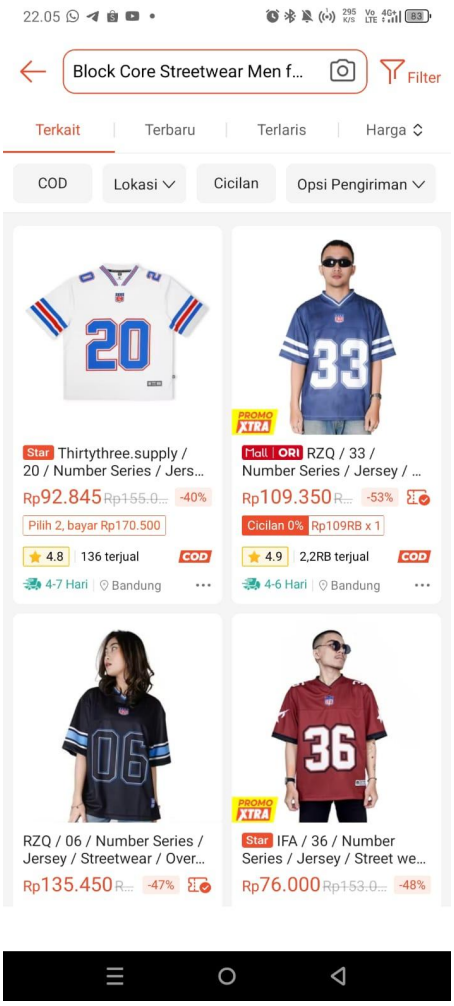
gaya streetwear modern yang lahir dari semangat bereksperimen dan mengaburkan batas antara dunia olahraga dan fashion jalanan. Keunikannya terletak pada penggunaan jersey bola sebagai item utama, menjadikannya lebih dari sekadar simbol fandom —melainkan pernyataan gaya yang berani. Gaya ini merepresentasikan generasi muda yang dinamis, penuh karakter, dan tidak takut memadukan elemen yang tampak kontras. Block Core bukan hanya soal pakaian, tapi juga tentang cara mengekspresikan diri dengan bebas dan orisinal.

Atasan: Jersey klub bola bergaya retro dengan motif kotak merah-putih dan logo "PACHA Ibiza" yang menonjol.

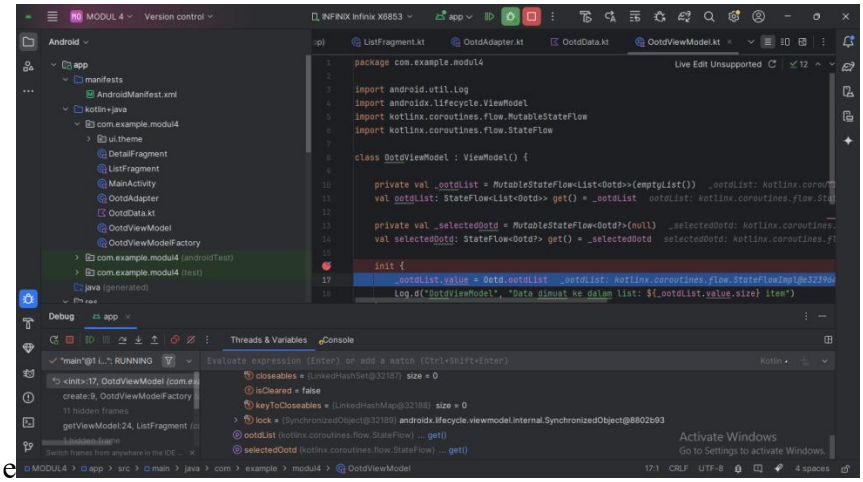
Bawahan: Celana wide leg bahan wol berwarna abu-abu tua, menciptakan kontras menarik antara sporty dan formal.

SHOP THIS STYLE

Gambar 16 Hasil Tampilan UI Detail Soal 1



Gambar 17 Screenshot E-commerce setelah Intent Soal 1



e

Gambar 18 Screenshot Debungging

C. Pembahasan

1. MainActivity.kt

File ini merupakan titik masuk utama aplikasi Android. Di dalamnya, kelas MainActivity mewarisi AppCompatActivity, yang merupakan salah satu komponen utama dalam Android untuk mendukung kompatibilitas UI lintas versi. Di dalam metode onCreate(), dipanggil setContentView(R.layout.activity_main), yang berarti aplikasi akan menampilkan tampilan (UI) dari activity_main.xml sebagai antarmuka awal. File ini secara struktur sangat sederhana, namun sangat penting karena menginisialisasi layout utama dan menjadi wadah fragment yang akan dimuat, seperti ListFragment dan DetailFragment. Biasanya MainActivity juga menjadi pengatur navigasi fragment menggunakan NavHostFragment jika menggunakan Jetpack Navigation, meskipun tidak eksplisit terlihat di file ini.

2. DetailFragment.kt

File ini adalah fragment yang bertanggung jawab menampilkan detail dari satu item OOTD (Outfit of The Day). DetailFragment menggunakan *View Binding* melalui FragmentDetailBinding untuk mengakses elemen UI secara langsung tanpa perlu findViewById, yang meningkatkan keamanan dan efisiensi kode.

Di dalam metode onViewCreated, fragment menerima argument ootdId yang dikirim dari fragment sebelumnya (kemungkinan ListFragment). ID ini digunakan untuk mengambil objek Ootd tertentu melalui metode Ootd.getById(ootdId). Setelah data OOTD diperoleh, informasi seperti judul,

jenis kelamin, gaya, deskripsi, dan gambar akan ditampilkan melalui komponen UI yang telah di-*bind*.

Menariknya, terdapat fitur interaktif berupa tombol btnShop, yang ketika ditekan akan membuat URL pencarian berdasarkan properti OOTD (judul, gaya, dan gender) ke situs Shopee. Proses ini dilakukan dengan menyusun searchQuery, kemudian mengenkripsi query-nya menggunakan URLEncoder, lalu membuat *Intent* untuk membuka browser ke URL pencarian tersebut. Ini adalah contoh integrasi sederhana antara aplikasi dan layanan eksternal.

Metode onDestroyView() digunakan untuk menghapus referensi binding agar tidak terjadi memory leak, mengikuti praktik standar pengelolaan memori dalam penggunaan View Binding pada fragment.

3. ListFragment.kt:

ListFragment adalah fragment yang menampilkan daftar semua OOTD yang tersedia dalam bentuk RecyclerView. Sama seperti DetailFragment, fragment ini menggunakan FragmentListBinding untuk efisiensi akses ke komponen UI.

Pada onViewCreated, diinisialisasi sebuah OotdAdapter yang menerima data dari Ootd.ootdList, yaitu daftar statis atau dinamis dari semua outfit. Adapter ini menggunakan pola listener interface (OnItemClickListener) yang memiliki dua aksi utama: onDetailClick() dan onShopClick(). Pada onDetailClick, fragment akan melakukan navigasi ke DetailFragment dengan membawa ootdId sebagai argumen menggunakan Jetpack Navigation. Sedangkan pada onShopClick, konsepnya sama seperti pada DetailFragment,

yaitu membuat query pencarian berdasarkan detail OOTD dan membuka Shopee melalui Intent ke browser.

RecyclerView dikonfigurasi dengan LinearLayoutManager untuk menampilkan daftar secara vertikal. Terdapat juga logika untuk menangani keadaan jika daftar kosong (ootdList.isEmpty()), yang akan menyembunyikan RecyclerView dan menampilkan emptyView, semacam teks atau ikon placeholder yang menunjukkan bahwa tidak ada data tersedia.

Seperti sebelumnya, metode onDestroyView() digunakan untuk mencegah memory leak dengan menghapus binding saat fragment dihancurkan.

4. OotdAdapter.kt

File OotdAdapter.kt merupakan salah satu komponen utama dalam arsitektur RecyclerView Adapter yang bertugas menghubungkan data outfit (OOTD) dengan tampilan daftar item di UI. Kelas OotdAdapter menerima dua parameter utama: ootdList, yaitu daftar objek Ootd, dan listener, yang merupakan antarmuka OnItemClickListener untuk menangani aksi klik dari pengguna. Adapter ini mewarisi RecyclerView.Adapter dan di dalamnya didefinisikan kelas OotdViewHolder yang bertugas untuk mem-bind setiap elemen data ke tampilan item yang sudah ditentukan dalam item_ootd.xml.

Pada metode bind(), setiap properti dari objek Ootd seperti title, gender, style, description, dan imageResId ditampilkan ke dalam elemen visual seperti TextView dan ImageView. Tombol btnDetail dan btnImdb (yang seharusnya secara konteks bisa diubah menjadi btnShop) masing-masing memiliki click listener yang memanggil fungsi dari OnItemClickListener. Ini menunjukkan

bahwa adapter ini juga menangani interaksi pengguna, seperti ketika ingin melihat detail atau menuju toko. Selain itu, adapter juga menerapkan konten deskriptif gambar (contentDescription) demi mendukung aksesibilitas.

Metode onCreateViewHolder() berfungsi untuk meng-inflate layout item dari file XML dan mengembalikan OotdViewHolder. Sedangkan onBindViewHolder() memanggil fungsi bind() untuk memasukkan data OOTD berdasarkan posisi. Metode getItemCount() cukup sederhana, mengembalikan jumlah total data yang akan ditampilkan.

5. **OotdData.kt**

File OotdData.kt berisi deklarasi data model Ootd dan data source statis yang mewakili berbagai gaya fashion. Kelas Ootd menggunakan anotasi data class yang secara otomatis menyediakan fungsi seperti toString(), equals(), dan copy(). Setiap objek Ootd memiliki properti: id, title, style, description, imageResId, dan gender. Informasi ini mewakili detail outfit termasuk judul tampilan, kategori gaya (misalnya Streetwear, Romantic, Alternative), deskripsi naratif yang mendalam, serta referensi ke resource gambar.

Objek-objek Ootd dikelompokkan ke dalam daftar statis ootdList yang berfungsi sebagai data dummy atau konten statis dalam aplikasi. Masing-masing entri menggambarkan gaya fashion spesifik, lengkap dengan narasi inspiratif serta penjelasan atasan dan bawahan, yang sangat berguna untuk menciptakan pengalaman pengguna yang imersif dan edukatif. Ada berbagai gaya seperti Block Core, Cottage Core, Grunge, Japanese Americana, Old Money, dan Y2K, yang semuanya menjelaskan elemen fashion yang populer pada segmennya.

Fungsi getById() disediakan untuk mencari OOTD berdasarkan ID-nya, sangat berguna ketika aplikasi perlu menampilkan detail item tertentu saat pengguna memilih dari daftar.

6. **OotdViewModel.kt**

File OotdViewModel.kt ini berisi class OotdViewModel yang merupakan turunan dari ViewModel. ViewModel ini bertugas sebagai jembatan antara UI dan data yang ditampilkan, supaya data tetap terjaga meskipun terjadi perubahan konfigurasi seperti rotasi layar. Di dalamnya terdapat dua properti MutableStateFlow yaitu _ootdList dan _selectedOotd. _ootdList menyimpan list data outfit (OOTD) yang nanti akan ditampilkan di UI, sedangkan _selectedOotd menyimpan data OOTD yang sedang dipilih. Untuk menjaga encapsulation, kedua variabel tersebut dibungkus dengan StateFlow agar hanya bisa diakses dari luar tanpa bisa diubah secara langsung. Pada bagian init, list OOTD langsung diisi dari Ootd.ootdList dan dicetak ke log jumlah datanya. Ada juga fungsi selectOotd() yang dipanggil saat user memilih salah satu OOTD, lalu log akan menampilkan item mana yang dipilih berdasarkan judul dan gender-nya. File ini menunjukkan implementasi reactive programming menggunakan Kotlin Flow dan ViewModel.

7. **OotdViewModelFactory.kt**

File OotdViewModelFactory.kt berisi class OotdViewModelFactory yang mengimplementasikan ViewModelProvider.Factory. Fungsi utamanya

adalah untuk membuat instance dari OotdViewModel. Factory ini digunakan saat kita ingin membuat ViewModel secara manual, terutama kalau ViewModel-nya butuh parameter khusus atau untuk menjaga fleksibilitas pembuatan ViewModel. Pada override method create, terdapat pengecekan apakah modelClass yang diminta sesuai dengan class OotdViewModel, jika iya maka akan dikembalikan instance-nya. Jika tidak cocok, maka akan dilempar exception IllegalArgumentException. Penggunaan ViewModelFactory ini penting untuk memisahkan logika pembuatan ViewModel dari UI, sehingga kode lebih rapi dan mudah diuji.

8. activity_main.xml

File activity_main.xml merupakan file tata letak utama (main layout) dari aplikasi yang mendefinisikan struktur dasar UI dengan menggunakan FrameLayout. Di dalamnya, terdapat komponen FragmentContainerView yang berfungsi sebagai host dari navigasi berbasis fragment. View ini menggunakan NavHostFragment dari Jetpack Navigation Component untuk mengelola transisi antar fragment berdasarkan navigation graph (nav_graph.xml).

Atribut app:defaultNavHost="true" memastikan bahwa fragment ini akan menjadi host utama untuk navigasi (misalnya saat menekan tombol back). Atribut app:navGraph="@navigation/nav_graph" menunjukkan file navigasi XML yang berisi arah dan struktur perpindahan fragment. Tata letak ini juga memiliki latar belakang yang ditentukan melalui warna @color/background, memberikan kesan visual yang konsisten di seluruh aplikasi.

9. fragment_detail.xml

File fragment_detail.xml adalah layout tampilan detail untuk setiap OOTD yang dipilih. Layout dibungkus dalam ScrollView, memungkinkan pengguna menggulir jika kontennya panjang. Di dalamnya terdapat ConstraintLayout yang mengatur elemen-elemen UI secara fleksibel dan responsif. Elemen-elemen yang ditampilkan mencakup ImageView (ootdImage) untuk gambar outfit, serta TextView untuk ootdTitle, ootdGender, ootdStyle, dan ootdDescription. Semua teks menggunakan TextAppearance untuk menjaga konsistensi desain dan tema aplikasi.

Di bagian akhir terdapat tombol btnShop, yang memungkinkan pengguna melakukan aksi lanjut seperti membuka halaman belanja atau eksternal link ke produk serupa. Tata letak ini memberikan pengalaman detail yang informatif dan estetis bagi pengguna untuk memahami outfit secara menyeluruh. Gaya desainnya bersih, rapi, dan memprioritaskan keterbacaan serta hirarki informasi.

10. fragment_list.xml

File fragment_list.xml merupakan layout XML untuk tampilan utama dari ListFragment, yaitu sebuah fragment yang menampilkan daftar item OOTD (Outfit of The Day) dalam aplikasi. Layout ini menggunakan ConstraintLayout sebagai root layout, yang memungkinkan setiap elemen UI diposisikan dengan fleksibel dan efisien berdasarkan constraint antar elemen. Di dalamnya terdapat dua elemen utama: RecyclerView dan TextView.

RecyclerView dengan ID @+id/recyclerView berfungsi untuk menampilkan daftar OOTD secara vertikal menggunakan LinearLayoutManager. Atribut clipToPadding="false" dan padding sebesar 8dp memastikan tampilan daftar lebih lega dan tidak terpotong. RecyclerView

ini akan diisi oleh adapter yang mengatur item-item OOTD. Sebagai fallback, TextView dengan ID @+id/emptyView disediakan dan akan ditampilkan ketika daftar kosong. TextView ini menampilkan pesan dari string resource @string/empty_list_message, dan secara default disembunyikan (android:visibility="gone"), hanya muncul saat diperlukan. Layout ini sangat mendukung prinsip *responsive UI*, karena semua elemen menggunakan constraint terhadap parent, memastikan kompatibilitas pada berbagai ukuran layar.

11. item_ootd.xml

File ini adalah layout dari setiap item OOTD yang akan ditampilkan dalam RecyclerView. Root layout-nya menggunakan CardView, yang memberikan efek bayangan dan sudut membulat, menciptakan tampilan yang lebih menarik dan modern. Selanjutnya, ConstraintLayout digunakan untuk mengatur tata letak elemen-elemen di dalam card tersebut.

Elemen utama dari layout ini adalah ImageView (ID: ootdImage) yang menampilkan gambar OOTD dengan tinggi tetap 150dp dan properti centerCrop agar gambar proporsional memenuhi area tampilan. Kemudian terdapat beberapa TextView, masing-masing untuk menampilkan judul outfit (ootdTitle), jenis kelamin target outfit (ootdGender), gaya berpakaian (ootdStyle), dan deskripsi singkat (ootdDescription). Deskripsi dibatasi maksimal dua baris dengan ellipsize="end" untuk menjaga tampilan tetap rapi.

Pada bagian bawah terdapat LinearLayout horizontal yang memuat dua tombol (Button). Tombol pertama (btnImdb) biasanya digunakan untuk mengarahkan ke halaman toko atau e-commerce, dan tombol kedua (btnDetail) digunakan untuk membuka detail lengkap dari OOTD yang dipilih. Masing-masing tombol memiliki gaya yang konsisten sesuai style

Widget.OOTDList.Button. Dengan struktur ini, tiap item OOTD tidak hanya informatif secara visual, tetapi juga interaktif, memungkinkan pengguna mengambil tindakan langsung.

12. nav_graph.xml

File nav_graph.xml adalah bagian dari Android Jetpack Navigation Component yang mendefinisikan struktur navigasi antar fragment dalam aplikasi. Di dalam file ini terdapat dua fragment utama: listFragment dan detailFragment. listFragment berfungsi sebagai *startDestination*, artinya ketika aplikasi dijalankan, pengguna akan langsung diarahkan ke fragment ini yang menampilkan daftar OOTD.

Dari listFragment, terdapat satu action menuju detailFragment, yaitu action_listFragment_to_detailFragment. Action ini memungkinkan pengguna berpindah ke tampilan detail dari item yang dipilih. Sementara itu, detailFragment didefinisikan dengan satu argument bernama ootdId bertipe integer, yang akan digunakan untuk mengambil dan menampilkan data detail berdasarkan ID yang dikirim dari fragment sebelumnya. Konfigurasi ini memungkinkan navigasi antar UI dilakukan dengan mudah dan aman, karena argument telah diketik secara eksplisit (type-safe).

13. jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out

Debugger itu kayak detektif buat ngubugin kode. Fungsinya buat ngecek alur program, ngelihat nilai variabel pas runtime, dan nyari di mana kesalahan terjadi. Jadi, kita bisa nge-pause kode di titik tertentu (breakpoint) terus ngelakuin investigasi.

Cara pake debugger di Android Studio:

1. **Set breakpoint**: Klik area sebelah kiri nomor baris (nanti muncul titik merah).
2. **Jalankan debug mode**: Klik tombol debug (yang ada icon kumbang).
3. **Pantau eksekusi**: Pas kode berhenti di breakpoint, kita bisa liat nilai variabel di panel *Variables* atau *Watches*.

Fitur penting debugger:

- **Step Into (F7)**: Masuk ke dalem fungsi/method buat ngecek isinya. Misal, kalo ada calculateTotal(), kita bisa masuk ke dalem fungsi itu.
- **Step Over (F8)**: Lanjut ke baris selanjutnya *tanpa* masuk ke fungsi. Berguna kalo kita gak perlu ngecek dalem fungsi itu.
- **Step Out (Shift + F8)**: Keluar dari fungsi sekarang dan balik ke pemanggilnya. Cocok kalo udah selesai ngecek suatu fungsi.

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

Application class di Android adalah kelas inti yang mewakili seluruh aplikasi dan hidup selama aplikasi berjalan. Fungsinya sebagai tempat inisialisasi komponen global (seperti database, library networking), menyimpan data yang perlu diakses di banyak tempat (misalnya token autentikasi), dan menangani callback siklus hidup aplikasi. Cara pakainya dengan membuat subclass dari android.app.Application lalu mendaftarkannya di AndroidManifest.xml. Contohnya, ketika menggunakan Room Database

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

A. Source Code

1. MainActivity.kt

Tabel 27 Source Code Jawaban Soal 1

```
1   package com.example.modul5
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5
6   class MainActivity : AppCompatActivity() {
7       override fun onCreate(savedInstanceState: Bundle?) {
8           super.onCreate(savedInstanceState)
9           setContentView(R.layout.activity_main)
10      }
11  }
```

2. DetailFragment.kt

Tabel 28 Source Code Jawaban Soal 1

```
1   package com.example.modul5.ui
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import android.widget.Toast
10  import androidx.core.view.isVisible
11  import androidx.fragment.app.Fragment
12  import androidx.fragment.app.activityViewModels
13  import androidx.lifecycle.lifecycleScope
14  import coil.load
15  import com.example.modul5.data.MovieDetails
16  import com.example.modul5.databinding.FragmentDetailBinding
17  import kotlinx.coroutines.flow.collectLatest
18  import kotlinx.coroutines.launch
19
20  class DetailFragment : Fragment() {
21
22      private var _binding: FragmentDetailBinding? = null
23      private val binding get() = _binding!!
24
25      private val viewModel: MovieViewModel by activityViewModels {
26          MovieViewModelFactory(requireActivity().application)
      }

      override fun onCreateView(
```

```
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentDetailBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        viewLifecycleOwner.lifecycleScope.launch {
            viewModel.movieDetails.collectLatest { details ->
                // Tampilkan atau sembunyikan loading
                binding.progressBar.isVisible = details == null
                binding.contentGroup.isVisible = details != null

                details?.let {
                    bindMovieDetails(it)
                }
            }
        }
    }

    private fun bindMovieDetails(details: MovieDetails) {
        with(binding) {

moviePoster.load("https://image.tmdb.org/t/p/w500${details.posterP
ath}") {
                crossfade(true)
            }
            movieTitle.text = details.title
            movieOverview.text = details.overview

            // Info tambahan
            releaseYearText.text = "Tahun Rilis:
${details.releaseDate?.substring(0, 4) ?: "N/A"}"

            val director = details.credits.crew.find { it.job ==
"Director" }
            directorText.text = "Sutradara: ${director?.name ?:
"N/A"}"

            val actors =
details.credits.cast.take(3).joinToString(", ") { it.name }
            actorsText.text = "Aktor: $actors"

            // Logika Tombol Trailer
```

```
            val trailer = details.videos.results.find { it.site ==
"YouTube" && it.type == "Trailer" }
            if (trailer != null) {
                playTrailerButton.visibility = View.VISIBLE
                playTrailerButton.setOnClickListener {
                    val intent = Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.youtube.com/watch?v=${trailer.key}"))
                    startActivity(intent)
                }
            } else {
                playTrailerButton.visibility = View.GONE
            }
        }
    }

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

```

3. ListFragment.kt

Tabel 29 Source Code Jawaban Soal 1

```
package com.example.modul5.ui

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.fragment.app.activityViewModels
import androidx.lifecycle.Observer // Import Observer secara
eksplisit
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.GridLayoutManager
import com.example.modul5.R
import com.example.modul5.data.Movie
import com.example.modul5.databinding.FragmentListBinding

class ListFragment : Fragment() {

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

    private val viewModel: MovieViewModel by activityViewModels {
        MovieViewModelFactory(requireActivity().application)
```

```
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentListBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        // Tentukan tipe lambda secara eksplisit
        val adapter = MovieAdapter { movie: Movie ->
            viewModel.selectMovie(movie)

findNavController().navigate(R.id.action_listFragment_to_detailFra
gment)
        }

        binding.recyclerView.layoutManager =
GridLayoutManager(requireContext(), 2)
        binding.recyclerView.adapter = adapter

        // Tentukan tipe data yang di-observe
        viewModel.movies.observe(viewLifecycleOwner,
Observer<List<Movie>> { movies ->
            // Gunakan submitList untuk ListAdapter
            adapter.submitList(movies)
        })
    }

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

4. MovieAdapter.kt

Tabel 30 Source Code Jawaban Soal 1

```
1  package com.example.modul5.ui
2
3  import android.view.LayoutInflater
```

```
4  import android.view.ViewGroup
5  import androidx.recyclerview.widget.DiffUtil
6  import androidx.recyclerview.widget.ListAdapter
7  import androidx.recyclerview.widget.RecyclerView
8  import coil.load
9  import com.example.modul5.data.Movie
10 import com.example.modul5.databinding.ItemMovieBinding

11 class MovieAdapter(private val onClick: (Movie) -> Unit) :
12     ListAdapter<Movie,
13 MovieAdapter.MovieViewHolder>(MovieDiffCallback) {
14
15     // Inner class ViewHolder
16     class MovieViewHolder(private val binding: ItemMovieBinding) :
17 RecyclerView.ViewHolder(binding.root) {
18         fun bind(movie: Movie, onClick: (Movie) -> Unit) {
19             binding.movieTitle.text = movie.title

20 binding.moviePoster.load("https://image.tmdb.org/t/p/w500${movie.p
21 osterPath}") {
22                 crossfade(true)
23                 placeholder(android.R.drawable.ic_menu_gallery)
24
25 error(android.R.drawable.ic_menu_close_clear_cancel)
26             }
27             // Gunakan binding.root untuk setOnClickListener
28             binding.root.setOnClickListener {
29                 onClick(movie)
30             }
31         }
32     }
33
34     override fun onCreateViewHolder(parent: ViewGroup, viewType:
35 Int): MovieViewHolder {
36         val binding =
37 ItemMovieBinding.inflate(LayoutInflater.from(parent.context),
38 parent, false)
39         return MovieViewHolder(binding)
40     }

41     override fun onBindViewHolder(holder: MovieViewHolder,
42 position: Int) {
43         val movie = getItem(position)
44         holder.bind(movie, onClick)
     }
45 }

46 // Object untuk DiffUtil di luar kelas Adapter
47 object MovieDiffCallback : DiffUtil.ItemCallback<Movie>() {
```

```
48      override fun areItemsTheSame(oldItem: Movie, newItem: Movie):
49  Boolean {
50          return oldItem.id == newItem.id
51      }

      override fun areContentsTheSame(oldItem: Movie, newItem:
    Movie): Boolean {
          return oldItem == newItem
      }
    }
```

5. Movie.kt

Tabel 31 Source Code Jawaban Soal 1

```
1   @file:OptIn(InternalSerializationApi::class)
2
3   package com.example.modul5.data
4
5   import androidx.room.Entity
6   import androidx.room.PrimaryKey
7   import kotlinx.serialization.InternalSerializationApi
8   import kotlinx.serialization.SerialName
9   import kotlinx.serialization.Serializable
10  import kotlin.OptIn
11
12  // Kelas untuk daftar film (tetap sama)
13  @Serializable
14  @Entity(tableName = "movies")
15  data class Movie(
16      @PrimaryKey
17      val id: Int,
18      val title: String,
19      @SerialName("poster_path")
20      val posterPath: String?,
21      @SerialName("overview")
22      val overview: String
23  )
24
25  // Wrapper untuk daftar film populer
26  @Serializable
27  data class MovieResponse(
28      val results: List<Movie>
29  )
30
31  //--- KELAS-KELAS BARU UNTUK DETAIL FILM ---
32
33  // Kelas utama untuk detail film
```

```
@Serializable
data class MovieDetails(
    val id: Int,
    val title: String,
    val overview: String,
    @SerialName("poster_path")
    val posterPath: String?,
    @SerialName("release_date")
    val releaseDate: String?,
    val credits: Credits,
    val videos: VideoResponse
)

// Kelas untuk kredit (aktor dan kru)
@Serializable
data class Credits(
    val cast: List<CastMember>,
    val crew: List<CrewMember>
)

// Kelas untuk anggota cast (aktor)
@Serializable
data class CastMember(
    val name: String,
    val character: String
)

// Kelas untuk anggota kru (sutradara, dll)
@Serializable
data class CrewMember(
    val name: String,
    val job: String
)

// Wrapper untuk daftar video/trailer
@Serializable
data class VideoResponse(
    val results: List<Video>
)

// Kelas untuk satu video/trailer
@Serializable
data class Video(
    val key: String,
    val site: String,
    val type: String
)
```

6. MovieView Model.kt

Tabel 32 Source Code Jawaban Soal 1

```
package com.example.modul5.ui

import android.app.Application
import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.asLiveData
import androidx.lifecycle.viewModelScope
import com.example.modul5.data.AppDatabase
import com.example.modul5.data.Movie
import com.example.modul5.data.MovieDetails
import com.example.modul5.data.MovieRepository
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch

class MovieViewModel(application: Application) : ViewModel() {

    private val repository: MovieRepository
    val movies: LiveData<List<Movie>>

    // StateFlow untuk film yang dipilih dari daftar
    private val _selectedMovie = MutableStateFlow<Movie?>(null)
    val selectedMovie: StateFlow<Movie?> = _selectedMovie

    // --- STATEFLOW BARU UNTUK MENAMPUNG DETAIL LENGKAP ---
    private val _movieDetails = MutableStateFlow<MovieDetails?>(null)
    val movieDetails: StateFlow<MovieDetails?> = _movieDetails

    init {
        val movieDao = AppDatabase.getDatabase(application).movieDao()
        repository = MovieRepository(movieDao)
        movies = repository.movies.asLiveData()
        refreshDataFromRepository()
    }

    private fun refreshDataFromRepository() {
```

```
39          viewModelScope.launch {
40              repository.refreshMovies()
41          }
42      }
43
44
45      fun selectMovie(movie: Movie) {
46          _selectedMovie.value = movie
            // Saat film dipilih, langsung ambil detail
47  lengkapnya
48          fetchMovieDetails(movie.id)
49      }
50
        // --- FUNGSI BARU UNTUK MENGAMBIL DETAIL ---
51      private fun fetchMovieDetails(movieId: Int) {
52          viewModelScope.launch {
53              // Set null dulu agar UI menampilkan loading
            (jika ada)
54              _movieDetails.value = null
55              // Panggil repository untuk mendapatkan detail
56              val details =
57  repository.getMovieDetails(movieId)
58              _movieDetails.value = details
            }
        }
    }
```

7. MovieView ModelFactory.kt

Tabel 33 Source Code Jawaban Soal 1

```
1   package com.example.modul5.ui
2
3   import android.app.Application
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6
7
8   class MovieViewModelFactory(private val application:
9   Application) : ViewModelProvider.Factory {
10      override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
            if
```

```
11  (modelClass.isAssignableFrom(MovieViewModel::class.java))
    {
12              @Suppress("UNCHECKED_CAST")
13              return MovieViewModel(application) as T
            }
14          throw IllegalArgumentException("Unknown ViewModel
class")
15      }
    }
```

8. activity_main.xml

Tabel 34 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph" />
</FrameLayout>
```

9. Fragment_detail.xml

Tabel 35 Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#F5F5F5"
    tools:context=".ui.DetailFragment">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">

        <!-- Progress Bar untuk loading -->
        <ProgressBar
            android:id="@+id/progress_bar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="parent"
```

```
22              app:layout_constraintBottom_toBottomOf="parent"
23              app:layout_constraintStart_toStartOf="parent"
24              app:layout_constraintEnd_toEndOf="parent"
25              android:visibility="visible" />
26
27          <!-- Grup konten yang akan ditampilkan setelah loading
    selesai -->
28          <androidx.constraintlayout.widget.Group
29              android:id="@+id/content_group"
30              android:layout_width="wrap_content"
31              android:layout_height="wrap_content"
32              android:visibility="gone"
33
    app:constraint_referenced_ids="movie_poster,movie_title,play_tra
    iler_button,label_overview,movie_overview,label_info,info_contai
    ner"
34              tools:visibility="visible"/>
35
36          <com.google.android.material.card.MaterialCardView
37              android:id="@+id/movie_poster_card"
38              android:layout_width="150dp"
39              android:layout_height="225dp"
40              app:cardCornerRadius="12dp"
41              app:cardElevation="8dp"
42              app:layout_constraintStart_toStartOf="parent"
43              app:layout_constraintTop_toTopOf="parent">
44
45              <ImageView
46                  android:id="@+id/movie_poster"
47                  android:layout_width="match_parent"
48                  android:layout_height="match_parent"
49                  android:scaleType="centerCrop"
50                  tools:src="@tools:sample/avatars" />
51
52          </com.google.android.material.card.MaterialCardView>
53
54          <TextView
55              android:id="@+id/movie_title"
56              android:layout_width="0dp"
57              android:layout_height="wrap_content"
58              android:layout_marginStart="16dp"
59
    android:textAppearance="?attr/textAppearanceHeadline6"
60              android:textColor="@android:color/black"
61              android:textStyle="bold"
62              app:layout_constraintEnd_toEndOf="parent"
63
    app:layout_constraintStart_toEndOf="@id/movie_poster_card"
64
```

```
    app:layout_constraintTop_toTopOf="@id/movie_poster_card"
65              tools:text="Judul Film yang Sangat Panjang" />
66
67          <Button
68              android:id="@+id/play_trailer_button"
69              style="@style/Widget.MaterialComponents.Button.Icon"
70              android:layout_width="160dp"
71              android:layout_height="100dp"
72              android:text="Tonton Trailer"
73              app:icon="@drawable/ic_play_arrow"
74
    app:layout_constraintBottom_toBottomOf="@id/movie_poster_card"
75
    app:layout_constraintStart_toStartOf="@id/movie_title" />
76
77          <TextView
78              android:id="@+id/label_info"
79              android:layout_width="wrap_content"
80              android:layout_height="wrap_content"
81              android:text="Informasi"
82              android:layout_marginTop="24dp"
83
    android:textAppearance="?attr/textAppearanceTitleMedium"
84              android:textSize="20sp"
85              android:textStyle="bold"
86              android:textColor="@android:color/black"
87              app:layout_constraintStart_toStartOf="parent"
88
    app:layout_constraintTop_toBottomOf="@id/movie_poster_card"/>
89
90          <LinearLayout
91              android:id="@+id/info_container"
92              android:layout_width="0dp"
93              android:layout_height="wrap_content"
94              android:orientation="vertical"
95              android:layout_marginTop="8dp"
96              android:padding="12dp"
97              android:background="@drawable/rounded_background"
98              app:layout_constraintTop_toBottomOf="@id/label_info"
99              app:layout_constraintStart_toStartOf="parent"
100             app:layout_constraintEnd_toEndOf="parent">
101
102             <TextView android:id="@+id/release_year_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" tools:text="Tahun Rilis:
    2025" android:textColor="@android:color/black"/>
103             <TextView android:id="@+id/director_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" tools:text="Sutradara: John
```

```
      Doe" android:layout_marginTop="4dp"
      android:textColor="@android:color/black"/>
104           <TextView android:id="@+id/actors_text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" tools:text="Aktor: A, B, C"
      android:layout_marginTop="4dp"
      android:textColor="@android:color/black"/>
105
106       </LinearLayout>
107
108       <TextView
109           android:id="@+id/label_overview"
110           android:layout_width="wrap_content"
111           android:layout_height="wrap_content"
112           android:text="Sinopsis"
113           android:layout_marginTop="16dp"
114
      android:textAppearance="?attr/textAppearanceTitleMedium"
115           android:textSize="20sp"
116           android:textStyle="bold"
117           android:textColor="@android:color/black"
118           app:layout_constraintStart_toStartOf="parent"
119
      app:layout_constraintTop_toBottomOf="@id/info_container"/>
120
121       <TextView
122           android:id="@+id/movie_overview"
123           android:layout_width="0dp"
124           android:layout_height="wrap_content"
125           android:layout_marginTop="8dp"
126           android:textAppearance="?attr/textAppearanceBody2"
127           android:textColor="@android:color/black"
128           app:layout_constraintEnd_toEndOf="parent"
129           app:layout_constraintStart_toStartOf="parent"
130
      app:layout_constraintTop_toBottomOf="@id/label_overview"
131           tools:text="Ini adalah deskripsi film yang sangat
      panjang..." />
132
          </androidx.constraintlayout.widget.ConstraintLayout>
133   </ScrollView>
```

10. Fragment_list.xml

Tabel 36 Source Code Jawaban Soal 1

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

```
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@color/background">
8
9       <androidx.recyclerview.widget.RecyclerView
10          android:id="@+id/recyclerView"
11          android:layout_width="match_parent"
12          android:layout_height="match_parent"
13          android:clipToPadding="false"
14          android:padding="8dp"

15  app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

16  android:contentDescription="@string/recycler_content_description"
17          app:layout_constraintBottom_toBottomOf="parent"
18          app:layout_constraintEnd_toEndOf="parent"
19          app:layout_constraintStart_toStartOf="parent"
20          app:layout_constraintTop_toTopOf="parent" />
21
22      <TextView
23          android:id="@+id/emptyView"
24          android:layout_width="match_parent"
25          android:layout_height="500dp"
26          android:gravity="center"
27          android:text="@string/empty_list_message"
28          android:visibility="gone"
29          app:layout_constraintBottom_toBottomOf="parent"
30          app:layout_constraintEnd_toEndOf="parent"
31          app:layout_constraintStart_toStartOf="parent"
32          app:layout_constraintTop_toTopOf="parent" />
33  </androidx.constraintlayout.widget.ConstraintLayout>
```

11. item_movie.xml

Tabel 37 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <com.google.android.material.card.MaterialCardView
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="wrap_content"
8       android:layout_margin="8dp"
```

```
9       app:cardCornerRadius="12dp"
10      app:cardElevation="4dp">
11
12      <LinearLayout
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
15          android:orientation="vertical">
16
17          <ImageView
18              android:id="@+id/movie_poster"
19              android:layout_width="match_parent"
20              android:layout_height="250dp"
                android:scaleType="centerCrop"
21
    android:contentDescription="@string/recycler_content_description"
22              tools:srcCompat="@tools:sample/backgrounds/scenic" />
23
24          <TextView
25              android:id="@+id/movie_title"
26              android:layout_width="match_parent"
27              android:layout_height="wrap_content"
28              android:padding="12dp"
29
    android:textAppearance="?attr/textAppearanceTitleMedium"
30              android:gravity="center"
31              android:maxLines="2"
32              android:ellipsize="end"
33              android:textStyle="bold"
34              android:textColor="@color/white"
35              android:textSize="20sp"
36              tools:text="Judul Film yang Sangat Panjang Sekali
    Contohnya" />
37
38      </LinearLayout>
39
40  </com.google.android.material.card.MaterialCardView>
```

12. nav_graph.xml

Tabel 38 Source Code Jawaban Soal 1

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/nav_graph"
6       app:startDestination="@id/listFragment">
```

```
7
8       <fragment
9           android:id="@+id/listFragment"
10          android:name="com.example.modul5.ui.ListFragment"
11          android:label="Popular Movies"
12          tools:layout="@layout/fragment_list" >
13      <action
14          android:id="@+id/action_listFragment_to_detailFragment"
15          app:destination="@id/detailFragment" />
16      </fragment>
17
18      <fragment
19          android:id="@+id/detailFragment"
20          android:name="com.example.modul5.ui.DetailFragment"
21          android:label="Movie Detail"
22          tools:layout="@layout/fragment_detail" />
23
24  </navigation>
```

13. AppDatabase.kt

Tabel 39 Source Code Jawaban Soal 1

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

```
25              }
26          }
27      }
28  }
```

14. MovieDao.kt

Tabel 40 Source Code Jawaban Soal 1

```
1   package com.example.modul5.data
2
3   import androidx.room.Dao
4   import androidx.room.Insert
5   import androidx.room.OnConflictStrategy
6   import androidx.room.Query
7   import kotlinx.coroutines.flow.Flow
8
9   @Dao
10  interface MovieDao {
11      @Query("SELECT * FROM movies")
12      fun getAllMovies(): Flow<List<Movie>>
13
14      @Insert(onConflict = OnConflictStrategy.REPLACE)
15      suspend fun insertAll(movies: List<Movie>)
16
17      @Query("DELETE FROM movies")
18      suspend fun deleteAll()
19  }
```

15. MovieRepository.kt

Tabel 41 Source Code Jawaban Soal 1

```
1   package com.example.modul5.data
2
3   import android.util.Log
4   import com.example.modul5.networking.ApiClient
5   import kotlinx.coroutines.flow.Flow
6
7   class MovieRepository(private val movieDao: MovieDao) {
8
9       val movies: Flow<List<Movie>> = movieDao.getAllMovies()
10
11      suspend fun refreshMovies() {
12          try {
13              val response = ApiClient.instance.getPopularMovies()
```

```
14              movieDao.deleteAll()
15              movieDao.insertAll(response.results)
16          } catch (e: Exception) {
17              Log.e("MovieRepository", "Error refreshing movies:
    ${e.message}")
18          }
19      }
20
21      // --- FUNGSI BARU UNTUK MENGAMBIL DETAIL DARI INTERNET ---
22      suspend fun getMovieDetails(movieId: Int): MovieDetails? {
23          return try {
24              ApiClient.instance.getMovieDetails(movieId)
25          } catch (e: Exception) {
26              Log.e("MovieRepository", "Error getting movie
27  details: ${e.message}")
                null
28          }
29      }
30  }
```

16. ApiClient.kt

Tabel 42 Source Code Jawaban Soal 1

```
1   // File: networking/ApiClient.kt
2   package com.example.modul5.networking
3
4   import
    com.jakewharton.retrofit2.converter.kotlinx.serialization.asConvert
    erFactory
5   import kotlinx.serialization.json.Json
6   import okhttp3.Interceptor
7   import okhttp3.MediaType.Companion.toMediaType
8   import okhttp3.OkHttpClient
9   import retrofit2.Retrofit
10
11  object ApiClient {
12      private const val BASE_URL = "https://api.themoviedb.org/3/"
13
14      // PENTING: Masukkan API Read Access Token Anda di sini
        private const val API_READ_ACCESS_TOKEN =
    "eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIxMzY5MDE0ZDJjYjkxNjIzOWY1NjdkMWUwZ
    WZmZDUzNiIsIm5iZiI6MTc0OTcwNjcyNi4yMiwic3ViIjoiNjg0YTY3ZTYyNGMzOWVm
    NGU1MWViYTIyIiwic2NvcGVzIjpbImFwaV9yZWFkIl0sInZlcnNpb24iOjF9.Aojt3C
    Y11RJ5kjNKHHHGxKedOWNHzbvnqJ_1LBJ34TY"
15
16      private val json = Json {
17          ignoreUnknownKeys = true
18      }
19
20      val instance: ApiService by lazy {
```

```
21          // 1. Buat Interceptor untuk menambahkan header secara
    otomatis
22          val authInterceptor = Interceptor { chain ->
23              val originalRequest = chain.request()
24              val newRequest = originalRequest.newBuilder()
25                  .header("Authorization", "Bearer
    $API_READ_ACCESS_TOKEN")
26                  .header("accept", "application/json")
27                  .build()
28              chain.proceed(newRequest)
29          }
30
31          // 2. Buat OkHttpClient dan tambahkan interceptor
32          val okHttpClient = OkHttpClient.Builder()
33              .addInterceptor(authInterceptor)
34              .build()
35
36          // 3. Buat instance Retrofit dengan OkHttpClient yang sudah
    dimodifikasi
37          val retrofit = Retrofit.Builder()
38              .baseUrl(BASE_URL)
39              .client(okHttpClient) // Gunakan client custom kita
40              .addConverterFactory(json.asConverterFactory("applicati
41  on/json".toMediaType()))
42              .build()
43
44          retrofit.create(ApiService::class.java)
45      }
46  }
```

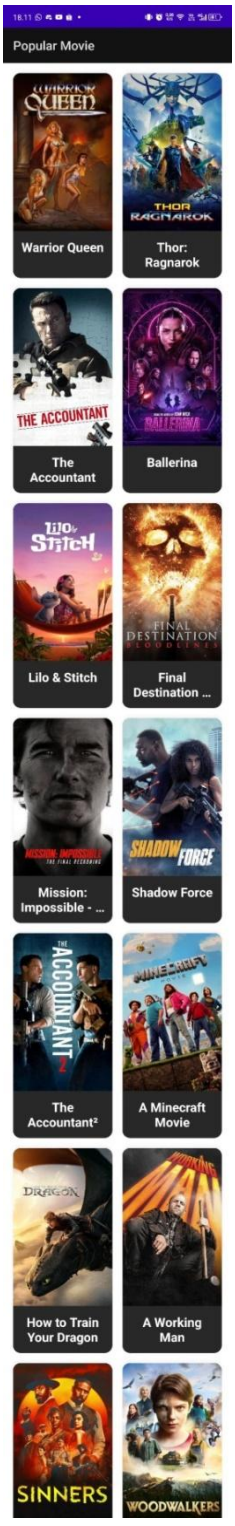
17. ApiService.kt

Tabel 43 Source Code Jawaban Soal 1

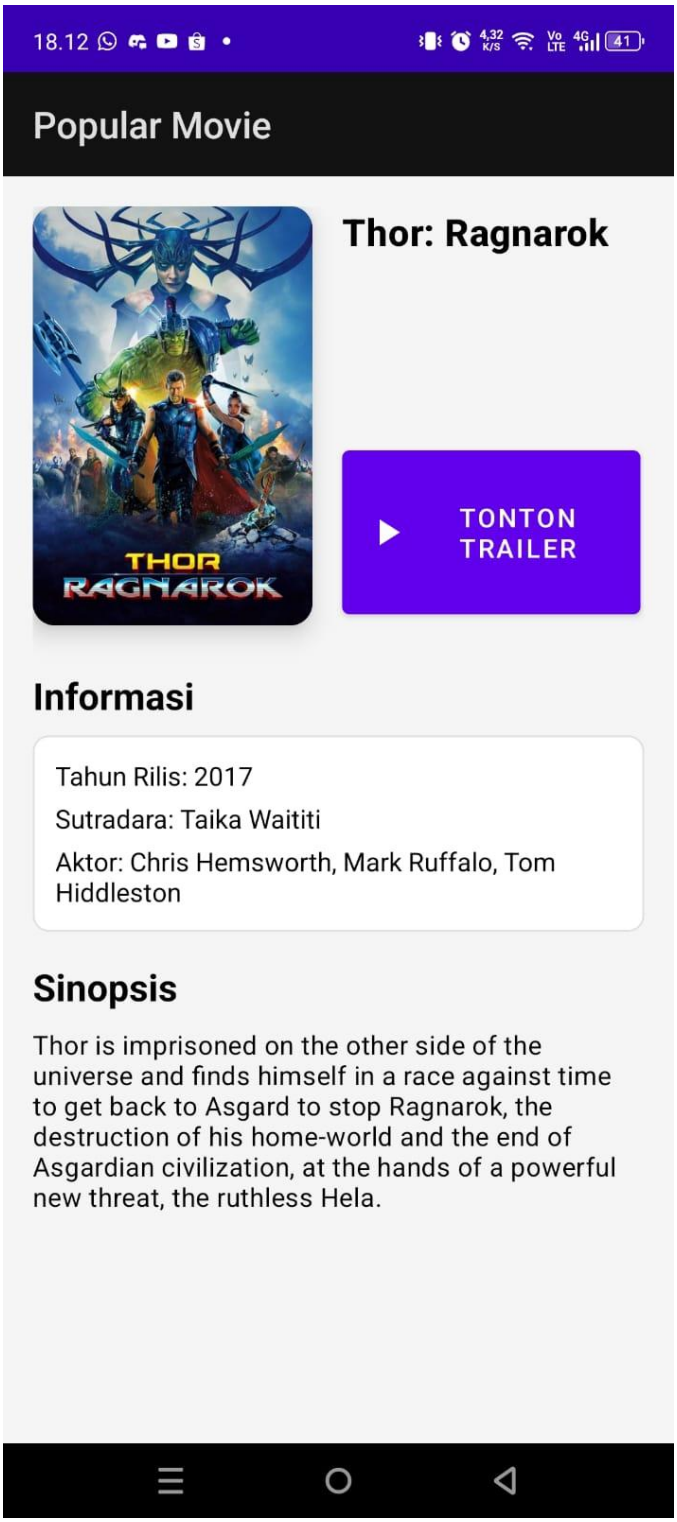
```
1   package com.example.modul5.networking
2
3   import com.example.modul5.data.MovieDetails
4   import com.example.modul5.data.MovieResponse
5   import retrofit2.http.GET
6   import retrofit2.http.Path
7   import retrofit2.http.Query
8
9   interface ApiService {
10      // Endpoint untuk daftar film populer (tetap sama)
11      @GET("movie/popular")
12      suspend fun getPopularMovies(
13          @Query("language") language: String = "en-US",
14          @Query("page") page: Int = 1
15      ): MovieResponse
```

```
16
17      // --- FUNGSI BARU UNTUK MENDAPATKAN DETAIL FILM ---
18      @GET("movie/{movie_id}")
19      suspend fun getMovieDetails(
20          @Path("movie_id") movieId: Int,
21          // 'append_to_response' adalah trik API TMDB untuk
    mendapatkan
22          // data kredit (aktor) dan video (trailer) dalam satu
    panggilan
23          @Query("append_to_response") appendToResponse: String =
    "videos,credits"
24      ): MovieDetails
25  }
```

B. Output Program



Gambar 19 Hasil Tampilan UI List Soal 1



Gambar 20 Hasil Tampilan UI Detail Soal 1

C. Pembahasan

1. MainActivity.kt

MainActivity adalah titik masuk utama aplikasi. Fungsinya sangat sederhana namun krusial, yaitu sebagai *host* untuk semua *fragment* dalam aplikasi ini. Dalam metode onCreate(), ia hanya mengatur *layout* utama dari activity_main.xml [320]. *Layout* ini berisi sebuah FragmentContainerView yang berfungsi sebagai wadah untuk NavHostFragment, komponen dari Jetpack Navigation yang mengelola semua transisi antar *fragment* (seperti dari ListFragment ke DetailFragment) sesuai dengan yang didefinisikan di nav_graph.xml

2. DetailFragment.kt

DetailFragment menampilkan informasi rinci dari satu film yang dipilih. Ia menerima id film melalui Safe Args dari Jetpack Navigation. Sama seperti ListFragment, ia membuat ViewModel sendiri untuk mengambil detail film. Dalam onViewCreated, ia mengobservasi state dari ViewModel (yang kemungkinan memuat data film tunggal berdasarkan ID). Setelah data diterima, ia mengisi semua View yang relevan: judul (movieTitle), sinopsis (movieOverview), tanggal rilis (movieReleaseDate), dan rating (movieRating). Library Glide juga digunakan di sini untuk memuat gambar poster film ke ImageView di bagian atas layar.

3. ListFragment.kt:

ListFragment adalah layar utama yang menampilkan daftar film. Ia menggunakan View Binding untuk mengakses view. Di dalam onViewCreated, ia menginisialisasi MovieRepository dan

MovieViewModelFactory untuk membuat MovieViewModel. RecyclerView diatur dengan LinearLayoutManager dan MovieAdapter. Bagian terpenting adalah blok viewLifecycleOwner.lifecycleScope.launch, di mana ia mengobservasi viewModel.movies menggunakan collectLatest. Setiap kali ada data baru, data tersebut akan dikirim ke adapter (adapter.submitList(movies)). Fragment ini juga menangani navigasi ke DetailFragment saat sebuah film diklik, dengan mengirimkan id film yang dipilih melalui NavDirections.

4. **MovieAdapter.kt**

jembatan antara data film dan RecyclerView. Ia menggunakan ListAdapter yang merupakan subclass dari RecyclerView.Adapter yang dioptimalkan untuk daftar yang dapat berubah, berkat penggunaan DiffUtil. Class internal MovieViewHolder bertanggung jawab untuk binding data satu objek Movie ke tampilan dalam item_movie.xml. Di dalam fungsi bind(), ia mengatur judul film dan tanggal rilis ke TextView yang sesuai, dan yang paling penting, ia menggunakan library Glide untuk memuat gambar poster dari URL (movie.poster_path) ke dalam ImageView, lengkap dengan placeholder dan gambar error. Adapter ini juga menangani klik pada setiap item dengan meneruskan aksi klik ke lambda onItemClick yang diinisialisasi dari ListFragment.

5. **Movie.kt**

mendefinisikan model data utama aplikasi, yaitu **data class Movie**. *Class* ini berfungsi sebagai representasi dari satu entitas film dan dirancang untuk bekerja dengan *database* Room dan respons dari API. Setiap objek Movie memiliki properti seperti id (integer, sebagai kunci utama atau

@PrimaryKey), title (judul film), overview (sinopsis), poster_path (URL untuk gambar poster), release_date (tanggal rilis), dan vote_average (rata-rata rating). Anotasi **@Entity(tableName = "movies")** menandakan bahwa *class* ini adalah sebuah tabel dalam *database* Room dengan nama "movies". Selain itu, anotasi **@SerializedName** digunakan pada setiap properti untuk memetakan nama *field* dari JSON yang diterima dari API (misalnya, poster_path) ke nama properti di dalam *class* Kotlin, memastikan deserialisasi data dari jaringan berjalan dengan benar.

Fungsi getById() disediakan untuk mencari OOTD berdasarkan ID-nya, sangat berguna ketika aplikasi perlu menampilkan detail item tertentu saat pengguna memilih dari daftar.

6. **MovieViewModel.kt**

MovieViewModel adalah kelas yang bertanggung jawab untuk menyediakan data ke UI dan mempertahankan state-nya dari perubahan konfigurasi. Kelas ini mengambil MovieRepository sebagai dependensi pada konstruktornya. Properti utamanya adalah movies, yang merupakan StateFlow<List<Movie>>. Properti ini diinisialisasi dengan memanggil repository.getMovies() dan mengubahnya menjadi StateFlow menggunakan stateIn(). Ini memungkinkan Fragment untuk mengobservasi daftar film secara reaktif. Terdapat juga fungsi refreshMovies() yang dipanggil di dalam blok init. Fungsi ini diluncurkan dalam viewModelScope dan memanggil repository.refreshMovies() untuk memastikan data film diambil dari API saat ViewModel pertama kali dibuat. Penanganan error sederhana juga disertakan dalam blok try-catch.

7. **MovieViewModelFactory.kt**

MovieViewModelFactory adalah kelas pabrik yang krusial untuk membuat instance dari MovieViewModel. Karena MovieViewModel memiliki dependensi (MovieRepository), kita tidak bisa membiarkan sistem membuatnya secara otomatis. Factory ini mengimplementasikan ViewModelProvider.Factory dan mengambil MovieRepository sebagai parameter. Di dalam metode create(), ia memeriksa apakah modelClass yang diminta adalah MovieViewModel::class.java. Jika benar, ia akan mengembalikan instance baru dari MovieViewModel dengan menyuntikkan repository yang telah diterimanya. Pola ini sangat penting untuk Dependency Injection dan pengujian (testing).

8. activity_main.xml

kerangka dasar untuk `MainActivity`. Satu-satunya elemen penting di dalamnya adalah `<androidx.fragment.app.FragmentContainerView>`. Komponen ini dikonfigurasi sebagai `NavHostFragment` melalui atribut `android:name`. Atribut `app:navGraph="@navigation/nav_graph"` menghubungkannya dengan grafik navigasi aplikasi, dan `app:defaultNavHost="true"` menjadikannya sebagai target utama untuk navigasi dan penanganan tombol "kembali" sistem.

9. fragment_detail.xml

Layout ini mendesain layar detail film. Biasanya menggunakan <ScrollView> sebagai elemen akar agar konten bisa digulir jika tidak muat di layar. Di dalamnya, ConstraintLayout atau LinearLayout digunakan untuk menyusun berbagai elemen UI seperti ImageView besar di bagian atas untuk poster film, diikuti oleh beberapa TextView untuk menampilkan judul, rating, tanggal rilis, dan sinopsis film yang bisa jadi cukup panjang.

10. fragment_list.xml

layout untuk ListFragment. Komponen utamanya adalah <androidx.recyclerview.widget.RecyclerView>, yang akan diisi dengan daftar film. Selain itu, biasanya terdapat juga komponen lain seperti <ProgressBar> untuk menunjukkan status loading saat data sedang diambil dari jaringan, dan mungkin sebuah <TextView> yang berfungsi sebagai emptyView untuk memberitahu pengguna jika tidak ada data yang dapat ditampilkan.

11. item_movie.xml

File ini mendefinisikan tampilan untuk satu baris atau satu item di dalam RecyclerView pada ListFragment. Umumnya menggunakan CardView sebagai kontainer untuk memberikan tampilan yang rapi dengan bayangan dan sudut melengkung. Di dalam card, terdapat ImageView untuk menampilkan poster film dan beberapa TextView untuk menampilkan informasi ringkas seperti judul dan tanggal rilis film.

12. nav_graph.xml

File ini adalah jantung dari komponen Navigasi Jetpack. Ia secara visual dan deklaratif mendefinisikan semua tujuan navigasi (yaitu, fragment) dan aksi yang menghubungkannya. Di sini, <fragment android:id="@+id/listFragment"> didefinisikan sebagai tujuan awal (app:startDestination). Terdapat sebuah <action> di dalam listFragment yang mendefinisikan transisi ke <fragment android:id="@+id/detailFragment">. Pentingnya, detailFragment didefinisikan dengan sebuah <argument> (misalnya, movieId dengan tipe integer), yang memungkinkan pengiriman

data (ID film) dari ListFragment ke DetailFragment dengan cara yang aman dan terjamin tipenya (type-safe

13. **AppDatabase.kt**

mendefinisikan kelas abstrak yang mewarisi RoomDatabase. Kelas ini berfungsi sebagai "pemegang" utama database dan titik akses utama ke data yang tersimpan. Anotasi @Database digunakan untuk mengkonfigurasi database, dengan menyebutkan class Movie sebagai satu-satunya entity (entities = [Movie::class]) dan mengatur nomor versi database (version = 1). Di dalamnya, terdapat sebuah fungsi abstrak movieDao() yang mengembalikan instance dari MovieDao, sehingga bagian lain dari aplikasi (seperti Repository) dapat mengakses metode-metode query yang telah didefinisikan. Terdapat juga sebuah companion object yang mengimplementasikan pola Singleton untuk memastikan hanya ada satu instance AppDatabase yang dibuat di seluruh aplikasi, mencegah masalah konkurensi dan menjaga konsistensi data.

14. **MovieDao.kt**

adalah interface MovieDao (Data Access Object), yang merupakan komponen inti dari Room Persistence Library. Interface ini mendefinisikan semua operasi database yang dibutuhkan oleh aplikasi. Anotasi @Dao memberitahu Room bahwa ini adalah sebuah DAO. Di dalamnya, terdapat beberapa fungsi yang diberi anotasi sesuai fungsinya: @Query("SELECT * FROM movies") pada fungsi getMovies() untuk mengambil semua data film dari tabel sebagai Flow<List<Movie>>, yang memungkinkan UI untuk mengobservasi perubahan data secara real-time. Fungsi insertAll() dengan anotasi @Insert(onConflict = OnConflictStrategy.REPLACE) digunakan untuk menyimpan daftar film ke dalam database; jika film dengan id yang sama sudah ada, data lama akan diganti dengan yang baru. Terakhir,

@Query("DELETE FROM movies") pada fungsi deleteAll() menyediakan cara untuk menghapus semua data dari tabel.

15. **MovieRepository.kt**

MovieRepository adalah kelas yang berperan sebagai perantara antara sumber data (database lokal dan jaringan) dengan seluruh bagian aplikasi lainnya (terutama ViewModel). Ini adalah implementasi dari Repository Pattern dan berfungsi sebagai Single Source of Truth (Satu Sumber Kebenaran). Repository ini memiliki dependensi ke ApiService (untuk mengambil data dari jaringan) dan MovieDao (untuk mengakses database lokal). Fungsi utamanya adalah getMovies(), yang mengembalikan Flow<List<Movie>> langsung dari MovieDao. Metode refreshMovies() bertugas untuk mengambil data film terbaru dari API melalui apiService.getMovies(). Jika panggilan API berhasil, data baru tersebut akan disimpan ke database lokal dengan memanggil movieDao.deleteAll() terlebih dahulu lalu movieDao.insertAll(). Dengan cara ini, UI selalu menampilkan data dari database, sementara repository menangani pembaruan data di latar belakang.

16. **ApiClient.kt**

Object ApiClient bertanggung jawab untuk membuat dan mengkonfigurasi instance Retrofit yang akan digunakan di seluruh aplikasi. Ini adalah implementasi dari pola Singleton. Di dalamnya, sebuah base URL untuk API (misalnya, The Movie Database API) didefinisikan. Kemudian, instance Retrofit dibuat menggunakan Retrofit.Builder(), yang dikonfigurasi dengan URL dasar tersebut dan sebuah converter factory (GsonConverterFactory.create()). Converter ini bertugas untuk mengubah respons JSON dari API menjadi objek-objek Kotlin (data class) secara otomatis. Akhirnya, instance ApiService dibuat dengan memanggil

retrofit.create(ApiService::class.java), yang siap digunakan untuk melakukan panggilan jaringan.

17. **ApiService.kt**

Interface ApiService adalah tempat di mana semua endpoint API didefinisikan menggunakan Retrofit. Di sini, hanya ada satu fungsi, yaitu getMovies(), yang diberi anotasi @GET("movie/popular"). Anotasi ini memberitahu Retrofit untuk membuat sebuah permintaan HTTP GET ke endpoint movie/popular relatif terhadap URL dasar yang dikonfigurasi di ApiClient. Fungsi ini juga menyertakan parameter api_key melalui anotasi @Query, yang akan ditambahkan ke URL sebagai parameter kueri (misalnya, ...?api_key=YOUR_API_KEY). Fungsi ini dideklarasikan sebagai suspend fun, yang berarti ia dapat dipanggil dari dalam sebuah coroutine tanpa memblokir thread utama, dan akan mengembalikan sebuah objek MovieResponse (sebuah data class yang mungkin membungkus daftar film dari API).

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

https://github.com/Ibnuuuuuuuuuuuuuuuuuuuuuuuuuuuuu/PEMROGRAMAN-MOBILE