

**LAPORAN AKHIR PRAKTIKUM  
PEMROGRAMAN MOBILE**



**Oleh:**

**Firda Khoirunisa**

**NIM. 2310817220025**

**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 : Firda Khoirunisa  
NIM : 2310817220025

Menyetujui,  
Asisten Praktikum

Mengetahui,  
Dosen Penanggung Jawab Praktikum

Zulfa Auliya Akbar  
NIM. 2210817210026

Muti`a Maulida S.Kom M.T.I  
NIP. 19881027 201903 20 13

## DAFTAR ISI

LEMBAR PENGESAHAN.....	2
DAFTAR ISI .....	3
DAFTAR GAMBAR.....	5
DAFTAR TABEL .....	6
MODUL 1 : Android Basic with Kotlin.....	8
SOAL 1.....	8
A. Source Code .....	10
B. Output Program .....	13
C. Pembahasan.....	14
MODUL 2: Android Layout.....	16
SOAL 1.....	16
A. Source Code .....	18
B. Output Program .....	22
C. Pembahasan.....	23
MODUL 3: Build a Scrollable List .....	25
SOAL 1.....	25
A. Source Code .....	28
B. Output Program .....	37
C. Pembahasan.....	38
SOAL 2.....	43
A. Penjelasan.....	43
MODUL 4: ViewModel and Debugging.....	44
SOAL 1.....	44
A. Source Code .....	45
B. Output Program .....	56
C. Pembahasan.....	57
SOAL 2.....	63
A. Penjelasan.....	63
MODUL 5: Connect to the Internet.....	64

SOAL 1 .....	64
A. Source Code .....	64
B. Output Program .....	84
C. Pembahasan .....	86
Tautan Git .....	95

## **DAFTAR GAMBAR**

Gambar 1. Screenshot Hasil Jawaban Soal No. 1 Modul 1 .....	13
Gambar 2. Screenshot Hasil Jawaban Soal No. 1 Modul 1 .....	13
Gambar 3. Screenshot Hasil Jawaban Soal No. 1 Modul 2 .....	22
Gambar 4. Screenshot Hasil Jawaban Soal No. 1 Modul 2 .....	22
Gambar 5. Soal No. 1 Modul 3 .....	26
Gambar 6. Soal No. 1 Modul 3 .....	27
Gambar 7. Screenshot Hasil Jawaban Soal 1 .....	37
Gambar 8. Screenshot Hasil Jawaban Soal 1 .....	38
Gambar 9. Soal No. 1 Modul 4 .....	44
Gambar 10. Screenshot Hasil Jawaban Soal No. 1 Modul 4 .....	56
Gambar 11. Screenshot Hasil Jawaban Soal No. 1 Modul 4 .....	56
Gambar 12. Screenshot Hasil Jawaban Soal No. 1 Modul 4 .....	57
Gambar 13. Screenshot Hasil Jawaban Soal No. 1 Modul 5 .....	84
Gambar 14. Screenshot Hasil Jawaban Soal No. 1 Modul 5 .....	85
Gambar 15. Screenshot Hasil Jawaban Soal No. 1 Modul 5 .....	86

## DAFTAR TABEL

Tabel 1. Source Code Jawaban Soal No. 1 Modul 1 .....	11
Tabel 2. Source Code Jawaban Soal No. 1 Modul 1 .....	12
Tabel 3. Source Code Jawaban Soal No. 1 Modul 2 .....	19
Tabel 4. Source Code Jawaban Soal No. 1 Modul 2 .....	21
Tabel 5. Source Code Jawaban Soal No. 1 Modul 3 .....	28
Tabel 6. Source Code Jawaban Soal No. 1 Modul 3 .....	28
Tabel 7. Source Code Jawaban Soal No. 1 Modul 3 .....	29
Tabel 8. Source Code Jawaban Soal No. 1 Modul 3 .....	31
Tabel 9. Source Code Jawaban Soal No. 1 Modul 3 .....	32
Tabel 10. Source Code Jawaban Soal No. 1 Modul 3 .....	32
Tabel 11. Source Code Jawaban Soal No. 1 Modul 3 .....	34
Tabel 12. Source Code Jawaban Soal No. 1 Modul 3 .....	34
Tabel 13. Source Code Jawaban Soal No. 1 Modul 3 .....	36
Tabel 14. Source Code Jawaban Soal No. 1 Modul 4 .....	45
Tabel 15. Source Code Jawaban Soal No. 1 Modul 4 .....	45
Tabel 16. Source Code Jawaban Soal No. 1 Modul 4 .....	46
Tabel 17. Source Code Jawaban Soal No. 1 Modul 4 .....	48
Tabel 18. Source Code Jawaban Soal No. 1 Modul 4 .....	49
Tabel 19. Source Code Jawaban Soal No. 1 Modul 4 .....	50
Tabel 20. Source Code Jawaban Soal No. 1 Modul 4 .....	50
Tabel 21. Source Code Jawaban Soal No. 1 Modul 4 .....	51
Tabel 22. Source Code Jawaban Soal No. 1 Modul 4 .....	53
Tabel 23. Source Code Jawaban Soal No. 1 Modul 4 .....	53
Tabel 24. Source Code Jawaban Soal No. 1 Modul 4 .....	55
Tabel 25. Source Code Jawaban Soal No.1 Modul 5 .....	65
Tabel 26. Source Code Jawaban Soal No.1 Modul 5 .....	65
Tabel 27. Source Code Jawaban Soal No.1 Modul 5 .....	67
Tabel 28. Source Code Jawaban Soal No.1 Modul 5 .....	67
Tabel 29. Source Code Jawaban Soal No.1 Modul 5 .....	68

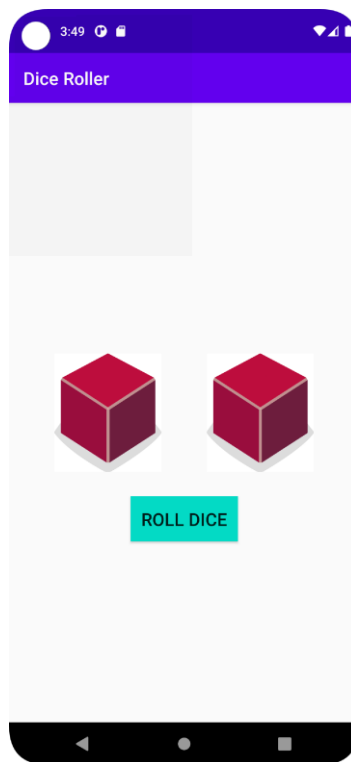
Tabel 30. Source Code Jawaban Soal No.1 Modul 5 .....	68
Tabel 31. Source Code Jawaban Soal No.1 Modul 5 .....	70
Tabel 32. Source Code Jawaban Soal No.1 Modul 5 .....	70
Tabel 33. Source Code Jawaban Soal No.1 Modul 5 .....	71
Tabel 34. Source Code Jawaban Soal No.1 Modul 5 .....	72
Tabel 35. Source Code Jawaban Soal No.1 Modul 5 .....	73
Tabel 36. Source Code Jawaban Soal No.1 Modul 5 .....	74
Tabel 37. Source Code Jawaban Soal No.1 Modul 5 .....	75
Tabel 38. Source Code Jawaban Soal No.1 Modul 5 .....	75
Tabel 39. Source Code Jawaban Soal No.1 Modul 5 .....	76
Tabel 40. Source Code Jawaban Soal No.1 Modul 5 .....	76
Tabel 41. Source Code Jawaban Soal No.1 Modul 5 .....	76
Tabel 42. Source Code Jawaban Soal No.1 Modul 5 .....	76
Tabel 43. Source Code Jawaban Soal No.1 Modul 5 .....	77
Tabel 44. Source Code Jawaban Soal No.1 Modul 5 .....	79
Tabel 45. Source Code Jawaban Soal No.1 Modul 5 .....	80
Tabel 46. Source Code Jawaban Soal No.1 Modul 5 .....	81
Tabel 47. Source Code Jawaban Soal No.1 Modul 5 .....	83

## MODUL 1 : Android Basic with Kotlin

### SOAL 1

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

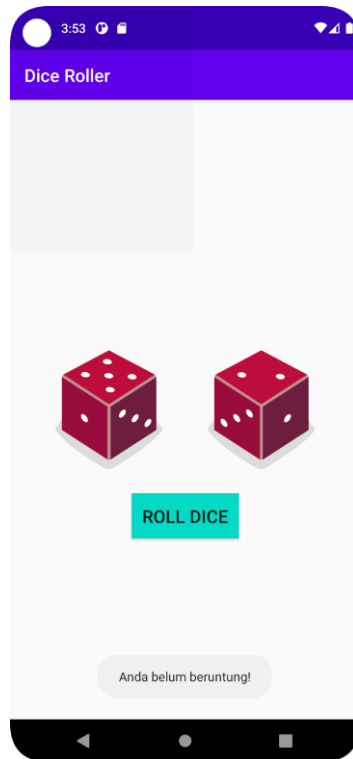
1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1.



**Gambar 1 Tampilan Awal Aplikasi**

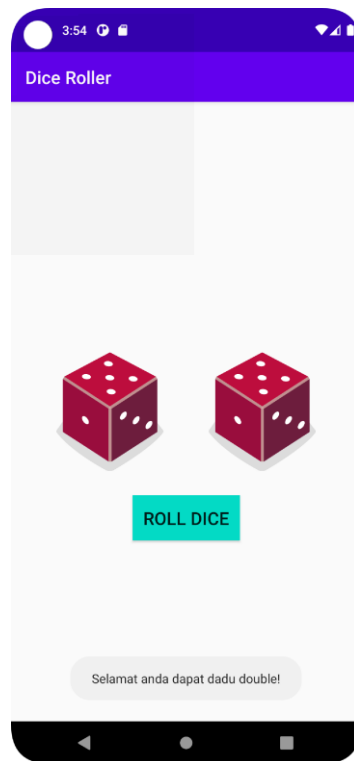
2. Setelah user menekan tombol “Roll Dice” maka masing-masing dadu akan memunculkan sisi dadu masing-masing dengan angka antara 1 s/d 6. Apabila user mendapatkan nilai dadu yang berbeda antara Dadu 1 dengan Dadu 2 maka akan menampilkan pesan “Anda belum beruntung!” seperti dapat dilihat pada Gambar 2.





**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=147HT2IIH5qin3z5ta7H9y2N\\_5OMW81Ll&export=download](https://drive.google.com/u/0/uc?id=147HT2IIH5qin3z5ta7H9y2N_5OMW81Ll&export=download)



**Gambar 3 Tampilan Roll Dadu Double**

## **A. Source Code**

### **1. MainActivity.kt**

```

1 package com.example.diceroller
2
3 import android.os.Bundle
4 import android.widget.Toast
5 import androidx.appcompat.app.AppCompatActivity
6 import com.example.diceroller.databinding.ActivityMainBinding
7
8 class MainActivity : AppCompatActivity() {
9     private lateinit var binding: ActivityMainBinding
10    private var dice1Value = 0
11    private var dice2Value = 0
12
13    override fun onCreate(savedInstanceState: Bundle?) {
14        super.onCreate(savedInstanceState)
15        binding = ActivityMainBinding.inflate(layoutInflater)
16        setContentView(binding.root)
17
18        binding.diceImage1.setImageResource(R.drawable.dice_0)
19        binding.diceImage2.setImageResource(R.drawable.dice_0)
20    }

```

```

21         savedInstanceState?.let {
22             dice1Value = it.getInt("DICE1")
23             dice2Value = it.getInt("DICE2")
24             updateDiceImages()
25         }
26
27         binding.buttonRoll.setOnClickListener {
28             rollDice()
29         }
30     }
31
32     private fun rollDice() {
33         dice1Value = (1..6).random()
34         dice2Value = (1..6).random()
35         updateDiceImages()
36
37         val message = if (dice1Value == dice2Value) {
38             "Selamat anda dapat dadu double!"
39         } else {
40             "Anda belum beruntung!"
41         }
42
43         Toast.makeText(this, message,
44             Toast.LENGTH_SHORT).show()
45     }
46
47     private fun updateDiceImages() {
48
49         binding.diceImage1.setImageResource(getDiceImage(dice1Value))
50
51         binding.diceImage2.setImageResource(getDiceImage(dice2Value))
52     }
53
54     private fun getDiceImage(value: Int): Int {
55         return when (value) {
56             1 -> R.drawable.dice_1
57             2 -> R.drawable.dice_2
58             3 -> R.drawable.dice_3
59             4 -> R.drawable.dice_4
60             5 -> R.drawable.dice_5
61             6 -> R.drawable.dice_6
62             else -> R.drawable.dice_0
63         }
64     }
65
66     override fun onSaveInstanceState(outState: Bundle) {
67         super.onSaveInstanceState(outState)
68         outState.putInt("DICE1", dice1Value)
69         outState.putInt("DICE2", dice2Value)
70     }
71 }

```

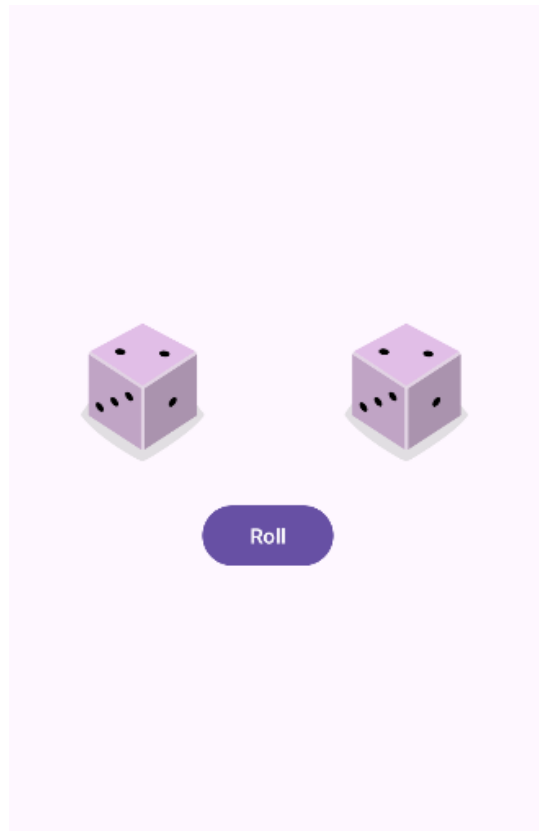
Tabel 1. Source Code Jawaban Soal No. 1 Modul 1

## 2. activity\_main.xml

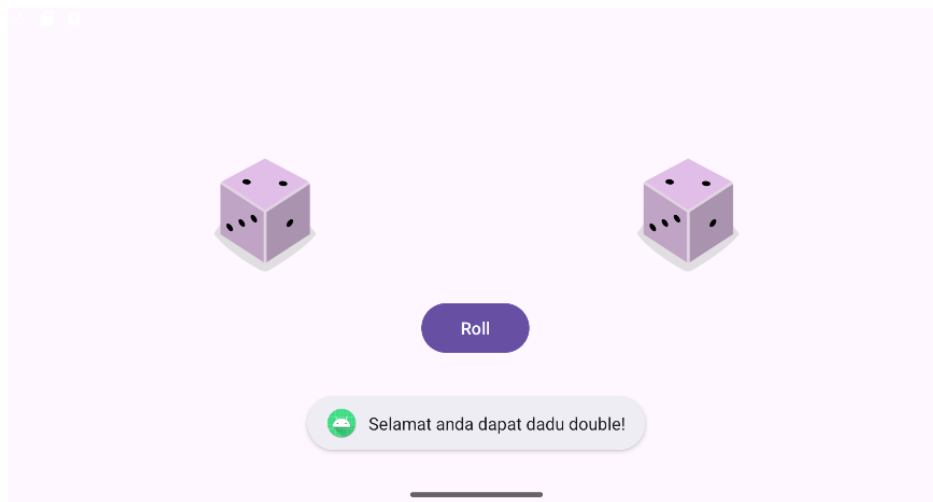
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10     <ImageView
11         android:id="@+id/dice_image_2"
12         android:layout_width="120dp"
13         android:layout_height="120dp"
14         android:layout_marginTop="392dp"
15         android:contentDescription="@string/dice_2"
16         android:src="@drawable/dice_0"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintStart_toEndOf="@+id/dice_image_1"
19         app:layout_constraintTop_toTopOf="parent" />
20
21     <ImageView
22         android:id="@+id/dice_image_1"
23         android:layout_width="120dp"
24         android:layout_height="120dp"
25         android:layout_marginTop="392dp"
26         android:contentDescription="@string/dice_1"
27         android:src="@drawable/dice_0"
28         app:layout_constraintEnd_toStartOf="@+id/dice_image_2"
29         app:layout_constraintStart_toStartOf="parent"
30         app:layout_constraintTop_toTopOf="parent" />
31
32     <Button
33         android:id="@+id/button_roll"
34         android:layout_width="wrap_content"
35         android:layout_height="wrap_content"
36         android:layout_marginTop="12dp"
37         android:text="@string/roll"
38         app:layout_constraintEnd_toEndOf="parent"
39         app:layout_constraintHorizontal_bias="0.486"
40         app:layout_constraintStart_toStartOf="parent"
41         app:layout_constraintTop_toBottomOf="@+id/dice_image_1"
42     />
43
44 </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 2. Source Code Jawaban Soal No. 1 Modul 1

## B. Output Program



Gambar 1. Screenshot Hasil Jawaban Soal No. 1 Modul 1



Gambar 2. Screenshot Hasil Jawaban Soal No. 1 Modul 1

## C. Pembahasan

### 1. MainActivity.kt:

Pada baris [3-6], mengimpor beberapa library penting, yaitu Bundle, Toast, dan AppCompatActivity, serta ActivityMainBinding untuk menghubungkan layout XML ke kode Kotlin. Kemudian, di baris [8 - 11], dideklarasikan kelas MainActivity yang mewarisi AppCompatActivity, serta dua variabel: binding sebagai penghubung layout dan dice1Value serta dice2Value untuk menyimpan nilai acak dadu. Pada baris [13-28], method onCreate() digunakan sebagai titik awal saat Activity dijalankan. Di dalamnya, binding diinisialisasi dengan memanggil ActivityMainBinding.inflate(layoutInflater) (baris 15), dan ditetapkan sebagai tampilan utama dengan setContentView(binding.root) pada baris 16. Setelah itu pada baris [18-19], gambar dadu diatur menggunakan gambar awal dice\_0. Jika ada data tersimpan dari instance sebelumnya, akan mengambil kembali nilai-nilai sebelumnya dengan getInt() lalu memperbarui gambar dadu menggunakan updateDiceImages(). Pada baris [27-28], mengatur listener pada tombol buttonRoll agar ketika tombol ditekan, metode rollDice() dipanggil. Method rollDice() didefinisikan di baris 32-44. Dalam metode ini, dua angka acak antara 1 dan 6 dihasilkan menggunakan (1..6).random() dan disimpan ke dalam dice1Value dan dice2Value (baris 33-34). Kemudian, gambar dadu diperbarui (baris 35) dan kondisi dicek—jika kedua dadu menunjukkan angka yang sama, akan muncul Toast dengan pesan keberuntungan; jika tidak, akan muncul pesan biasa (baris 37-40). Pada baris [47-51], metode updateDiceImages() mengatur ulang gambar pada kedua ImageView berdasarkan nilai dadu yang sekarang, dengan bantuan metode getDiceImage() yang akan mencocokkan angka dan mengembalikan resource drawable yang sesuai. Method getDiceImage() sendiri didefinisikan di baris [54-64]. Fungsi ini menerima value dan mencocokkannya dengan angka 1 hingga 6 untuk menentukan gambar yang sesuai (dice\_1 sampai dice\_6). Jika nilainya tidak sesuai, akan dikembalikan gambar dice\_0. Terakhir, pada baris [66-69], metode onSaveInstanceState() menyimpan nilai dadu terakhir (DICE1 dan DICE2) ke dalam Bundle agar dapat dipulihkan jika aplikasi di-restart atau orientasi layar berubah.

### 2. activity\_main.xml

Pada baris [2-8], struktur awal layout dideklarasikan menggunakan `ConstraintLayout`, yang memungkinkan kita mengatur posisi elemen UI secara fleksibel. Di baris [10-19], `ImageView` pertama (`dice_image_2`) dibuat dengan ukuran 120dp x 120dp dan menggunakan gambar awal `dice_0`. Elemen ini diletakkan di sebelah kanan `dice_image_1`, ditentukan oleh `constraint layout_constraintStart_toEndOf`. Di baris [21-30], elemen `ImageView` kedua (`dice_image_1`) didefinisikan. Posisinya berada di kiri dan diatur agar berdampingan dengan `dice_image_2`. Kedua gambar dadu ini diberi deskripsi konten dari string resource (`@string/dice_1` dan `@string/dice_2`) agar lebih ramah aksesibilitas. Baris [32-41] berisi elemen `Button` (`button_roll`) yang akan digunakan untuk memicu lemparan dadu. Tombol ini diatur agar berada di tengah-tengah layar secara horizontal (`layout_constraintHorizontal_bias`) dan berada tepat di bawah gambar dadu pertama. Dengan pendekatan layout ini, dua gambar dadu akan tampil berdampingan di bagian atas layar, sedangkan tombol "Roll" berada di bawahnya, siap ditekan oleh pengguna untuk menghasilkan angka acak.

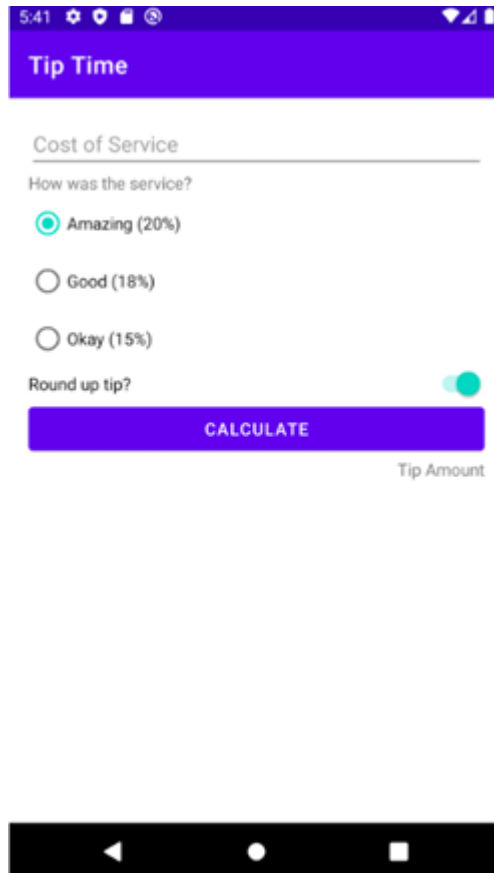
## **MODUL 2: Android Layout**

### **SOAL 1**

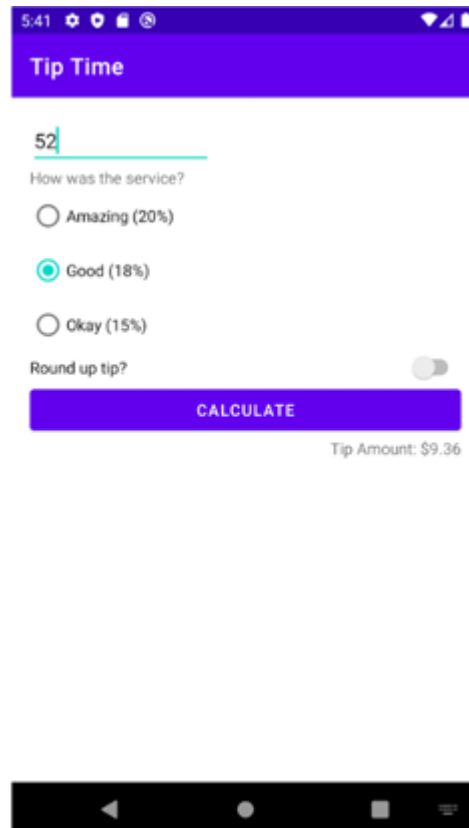
Buatlah sebuah aplikasi kalkulator tip yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

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





**Gambar 1 Tampilan Awal Aplikasi**



**Gambar 2 Tampilan Aplikasi Setelah Dijalankan**

## **A. Source Code**

### **1. MainActivity.kt**

```

1 package com.example.tipcalc
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import android.widget.Toast
6 import com.example.tipcalc.databinding.ActivityMainBinding
7 import java.text.NumberFormat
8 import kotlin.math.ceil
9
10 class MainActivity : AppCompatActivity() {
11     lateinit var binding: ActivityMainBinding
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         binding = ActivityMainBinding.inflate(layoutInflater)
16         setContentView(binding.root)
17         binding.calculateButton.setOnClickListener {
18

```

```

19 calculateTip()
20     }
21
22     private fun calculateTip() {
23         val inputText = binding.costOfService.text.toString()
24         val cost = inputText.toDoubleOrNull()
25
26         if (cost == null || cost == 0.0) {
27             binding.costOfService.error = "Masukkan angka yang
28 valid"
29             Toast.makeText(this, "Input tidak valid!",
30 Toast.LENGTH_SHORT).show()
31             binding.tipResult.text = ""
32             return
33         }
34
35         val selectedId =
36 binding.tipOptions.checkedRadioButtonId
37         val tipPersen = when (selectedId) {
38             R.id.option_ten_percent -> 0.2
39             R.id.option_seven_percent -> 0.18
40             else -> 0.15
41         }
42
43         var tip = cost * tipPersen
44         if (binding.roundTip.isChecked) {
45             tip = ceil(tip)
46         }
47
48         val currency =
49 NumberFormat.getCurrencyInstance().format(tip)
50         binding.tipResult.text = getString(R.string.tip_amount,
51 currency)
52     }
53
54     override fun onSaveInstanceState(outState: Bundle) {
55         super.onSaveInstanceState(outState)
56         outState.putString("tip_result",
57 binding.tipResult.text.toString())
58     }
59
60     override fun onRestoreInstanceState(savedInstanceState:
61 Bundle) {
62         super.onRestoreInstanceState(savedInstanceState)
63         val restoredTip =
64 savedInstanceState.getString("tip_result")
65         binding.tipResult.text = restoredTip
66     }
67 }
68

```

*Tabel 3. Source Code Jawaban Soal No. 1 Modul 2*

## 2. activity\_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4  xmlns:app="http://schemas.android.com/apk/res-auto"
5  xmlns:tools="http://schemas.android.com/tools"
6  android:id="@+id/main"
7  android:layout_width="match_parent"
8  android:layout_height="match_parent"
9  android:padding="16dp"
10  tools:context=".MainActivity">
11
12  <EditText
13      android:id="@+id/cost_of_service"
14      android:hint="Cost of Service"
15      android:inputType="number"
16      app:layout_constraintTop_toTopOf="parent"
17      app:layout_constraintLeft_toLeftOf="parent"
18      android:layout_width="match_parent"
19      android:layout_height="wrap_content" />
20
21  <TextView
22      android:id="@+id/services_question"
23      android:textColor="#CD7C7979"
24      android:layout_width="wrap_content"
25      android:layout_height="wrap_content"
26      app:layout_constraintStart_toStartOf="parent"
27
28  app:layout_constraintTop_toBottomOf="@id/cost_of_service"
29      android:text="How was the service?" />
30
31  <RadioGroup
32      android:id="@+id/tip_options"
33      android:orientation="vertical"
34      app:layout_constraintStart_toStartOf="parent"
35
36  app:layout_constraintTop_toBottomOf="@id/services_question"
37      android:layout_width="wrap_content"
38      android:layout_height="wrap_content">
39
40  <RadioButton
41      android:id="@+id/option_ten_percent"
42      android:text="Amazing (20%) "
43      android:layout_width="wrap_content"
44      android:layout_height="wrap_content"/>
45
46  <RadioButton
47      android:id="@+id/option_seven_percent"
48      android:text="Good (18%) "
49      android:layout_width="wrap_content"
50      android:layout_height="wrap_content"/>
```

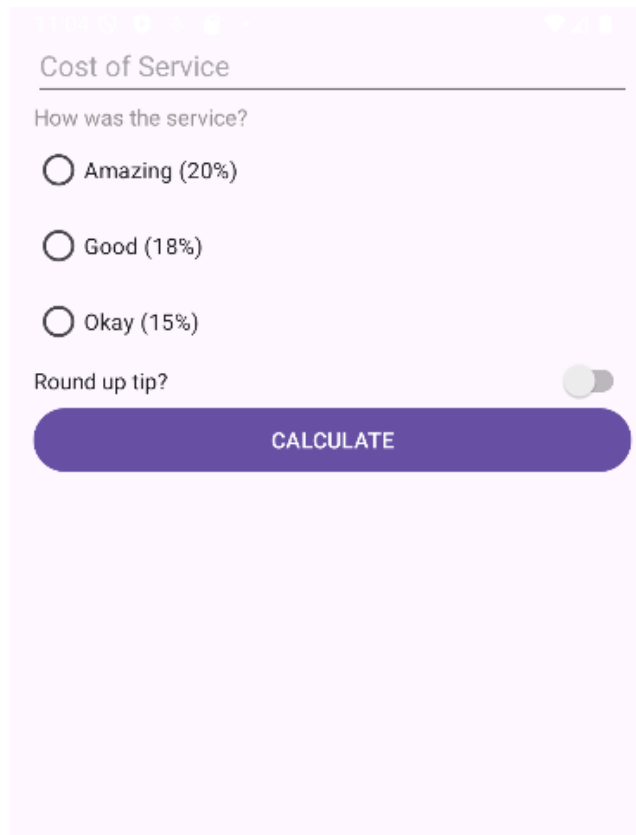
```

51
52         <RadioButton
53             android:id="@+id/option_five_percent"
54             android:text="Okay" (15%) "
55             android:layout_width="wrap_content"
56             android:layout_height="wrap_content"/>
57     </RadioGroup>
58
59     <Switch
60         android:id="@+id/round_tip"
61         android:checked="false"
62         app:layout_constraintTop_toBottomOf="@id/tip_options"
63         app:layout_constraintStart_toStartOf="@id/tip_options"
64         app:layout_constraintEnd_toEndOf="parent"
65         android:text="Round up tip?"
66         android:layout_width="0dp"
67         android:layout_height="wrap_content"/>
68
69     <Button
70         android:id="@+id/calculate_button"
71         app:layout_constraintStart_toStartOf="parent"
72         app:layout_constraintEnd_toEndOf="parent"
73         app:layout_constraintTop_toBottomOf="@id/round_tip"
74         android:text="CALCULATE"
75         android:layout_width="0dp"
76         android:layout_height="wrap_content"/>
77
78     <TextView
79         android:id="@+id/tip_result"
80         android:textColor="#CD7C7979"
81         tools:text="Tip Amount"
82
83     app:layout_constraintTop_toBottomOf="@id/calculate_button"
84     app:layout_constraintEnd_toEndOf="parent"
85     android:layout_width="wrap_content"
86     android:layout_height="wrap_content"/>
87
88 </androidx.constraintlayout.widget.ConstraintLayout>

```

*Tabel 4. Source Code Jawaban Soal No. 1 Modul 2*

## B. Output Program



Cost of Service

How was the service?

☐ Amazing (20%)

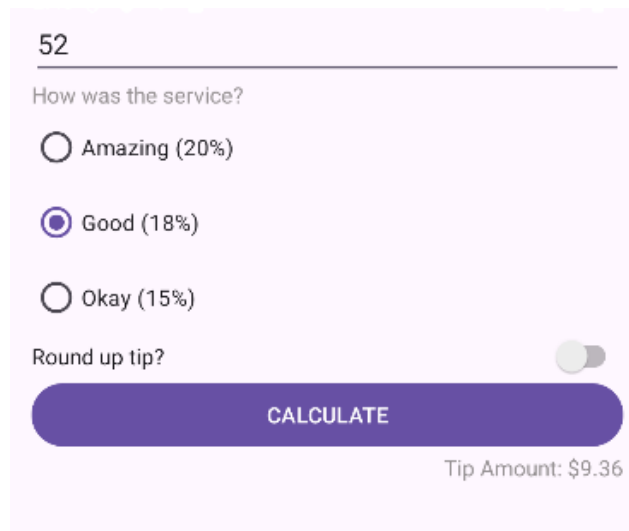
☐ Good (18%)

☐ Okay (15%)

Round up tip? ☐

CALCULATE

Gambar 3. Screenshot Hasil Jawaban Soal No. 1 Modul 2



52

How was the service?

☐ Amazing (20%)

☒ Good (18%)

☐ Okay (15%)

Round up tip? ☐

CALCULATE

Tip Amount: \$9.36

Gambar 4. Screenshot Hasil Jawaban Soal No. 1 Modul 2

## C. Pembahasan

### 1. MainActivity.kt:

Bagian ini merupakan deklarasi awal file. Baris [1] menunjukkan bahwa kelas MainActivity berada di dalam package com.example.tipcalc. Baris [3 – 8] adalah import dari kelas-kelas penting Android, seperti Bundle (untuk menyimpan data saat konfigurasi berubah), AppCompatActivity (kelas dasar untuk activity), dan Toast (untuk menampilkan pesan singkat). Baris [10] mendeklarasikan MainActivity sebagai turunan dari AppCompatActivity. Baris [12] mendeklarasikan properti binding dari tipe ActivityMainBinding, yang akan digunakan untuk mengakses elemen-elemen tampilan tanpa harus pakai findViewById. Baris [14 – 18] adalah onCreate(), yaitu metode yang dipanggil saat activity pertama kali dibuat. Baris [15] memanggil fungsi induk. Baris [16] menginisialisasi binding dengan cara meng-*inflate* layout activity\_main.xml. Baris [17] menetapkan tampilan yang digunakan (binding.root). Baris [18] menetapkan aksi ketika tombol hitung diklik, yaitu menjalankan fungsi calculateTip(). Baris [23 - 50] adalah fungsi calculateTip() yang berisi logika utama aplikasi. Baris [23 - 24] mengambil input dari pengguna dan mencoba mengubahnya menjadi angka (Double). Baris [26 - 29] memvalidasi input: jika kosong atau nol, maka muncul pesan error dan Toast serta hasil tip dikosongkan. Baris [35 - 40] menentukan persentase tip berdasarkan tombol radio yang dipilih. Baris 44 membulatkan tip jika pengguna mencentang opsi pembulatan (Switch). Baris [48] memformat hasil tip menjadi format mata uang. Baris [50] menampilkan hasilnya ke layar (TextView). Baris [54 - 56] Fungsi ini dijalankan ketika sistem akan menghancurkan activity, misalnya karena rotasi layar. Baris [56] menyimpan nilai tipResult ke dalam bundle agar bisa dipulihkan nanti. Baris [60 – 65] Fungsi ini dijalankan saat activity dipulihkan setelah dihancurkan, misal karena rotasi layar. Baris [63] mengambil kembali nilai hasil tip dari bundle dan menampilkannya di layar. Ini memastikan data tidak hilang saat orientasi layar berubah.

### 2. activity\_main.xml

Baris [1 – 4] menyatakan bahwa file menggunakan XML dan mendeklarasikan root view berupa `ConstraintLayout`, yaitu jenis layout yang fleksibel untuk mengatur posisi elemen berdasarkan constraint antar komponen. Baris [5 – 10] mengatur identitas layout (`@+id/main`) dan ukuran layout agar memenuhi seluruh layar (`match_parent`). `Padding` sebesar 16dp diberikan sebagai ruang tepi dalam. `tools:context` menunjukkan bahwa layout ini digunakan oleh `MainActivity`, hanya untuk keperluan preview di Android Studio. Pada baris [12 – 19], terdapat input teks untuk biaya layanan. `hint` menampilkan teks bantu saat kosong. `inputType="number"` membatasi input hanya angka. Constraint-nya ditempatkan di bagian atas dan kiri layout. Pada baris [21 – 29], `TextView` ini menampilkan pertanyaan tentang kualitas layanan. Letaknya berada di bawah `EditText`, dengan warna teks abu-abu (`#CD7C7979`). Pada baris [31 – 57] terdapat `RadioGroup` yang menampung tiga opsi tip dalam bentuk `RadioButton`. Orientasi `vertical`, dan default-nya adalah `option_seven_percent`. Setiap `RadioButton` merepresentasikan persentase tip berbeda: 20%, 18%, dan 15%. Pada baris [59 – 67], terdapat `Switch` yang memberikan opsi apakah tip harus dibulatkan ke atas. Awalnya tidak dicentang (`checked="false"`). Letaknya di bawah `RadioGroup` dan melebar ke kanan & kiri (`layout_width="0dp"` dengan constraint `horizontal`). Pada baris [69 – 76], terdapat Tombol yang digunakan untuk memicu perhitungan tip. Terletak di bawah `Switch`, memiliki teks "CALCULATE", dan melebar secara horizontal sesuai constraint. Pada baris [78 – 86], terdapat `TextView` yang digunakan untuk menampilkan hasil perhitungan tip. Warna teks abu-abu dan diisi sementara untuk preview dengan `tools:text="Tip Amount"`. Letaknya berada di bawah tombol hitung (`calculate_button`).



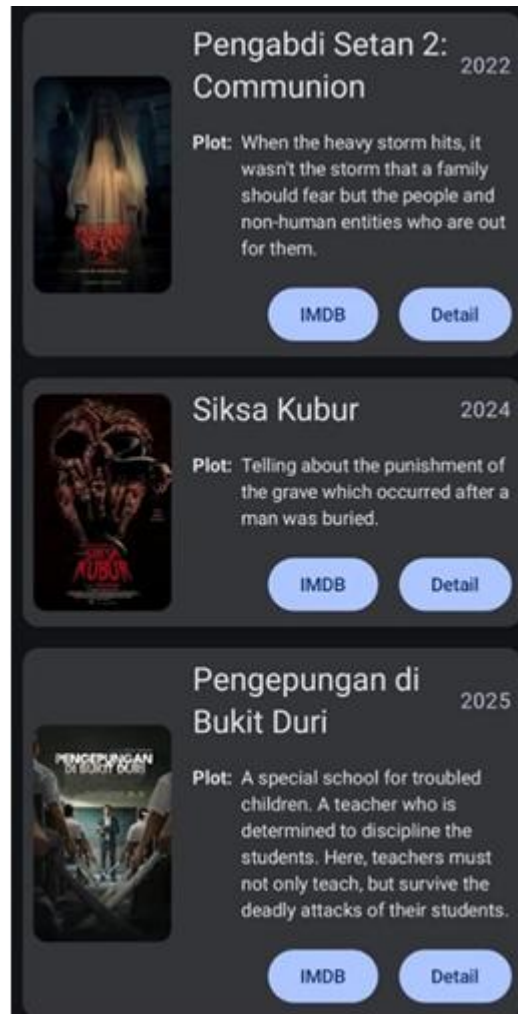
## **MODUL 3: Build a Scrollable List**

### **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 5. Soal No. 1 Modul 3

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 6. Soal No. 1 Modul 3*

## A. Source Code

### 1. MainActivity.kt

```
1 package com.example.modul3
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import com.example.modul3.databinding.ActivityMainBinding
6
7 class MainActivity : AppCompatActivity() {
8
9     private lateinit var binding: ActivityMainBinding
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         binding = ActivityMainBinding.inflate(layoutInflater)
14         setContentView(binding.root)
15
16         if (savedInstanceState == null) {
17             supportFragmentManager.beginTransaction()
18                 .replace(R.id.fragment_container,
19 HomeFragment())
20                 .commit()
21         }
22     }
23 }
```

Tabel 5. Source Code Jawaban Soal No. 1 Modul 3

### 2. Drama.kt

```
1 package com.example.modul3
2
3 import android.os.Parcelable
4 import kotlinx.parcelize.Parcelize
5
6 @Parcelize
7 data class Drama (
8     val title: String,
9     val year: String,
10    val genre: String,
11    val episodes: String,
12    val imageUrl: String,
13    val link: String
14 ) : Parcelable
```

Tabel 6. Source Code Jawaban Soal No. 1 Modul 3

### 3. ListDramaAdapter.kt

```
1 package com.example.modul3
2
```

```

3 import                                android.view.LayoutInflater
4 import                                android.view.ViewGroup
5 import                                androidx.recyclerview.widget.RecyclerView
6 import                                com.bumptech.glide.Glide
7 import                                com.example.modul3.databinding.ItemDramaBinding
8
9 class                                  ListDramaAdapter(
10     private val listDrama: List<Drama>,
11     private val onDetailClick: (Drama) -> Unit
12 ) : RecyclerView.Adapter<ListDramaAdapter.MyViewHolder>() {
13
14     inner class MyViewHolder(val binding: ItemDramaBinding) :
15     RecyclerView.ViewHolder(binding.root)
16
17     override fun onCreateViewHolder(parent: ViewGroup, viewType:
18     Int): MyViewHolder {
19         val binding =
20         ItemDramaBinding.inflate(LayoutInflater.from(parent.context),
21         parent, false)
22         return MyViewHolder(binding)
23     }
24
25     override fun getItemCount(): Int = listDrama.size
26
27     override fun onBindViewHolder(holder: MyViewHolder,
28     position: Int) {
29         val drama = listDrama[position]
30         holder.binding.drama = drama
31
32         Glide.with(holder.binding.root.context)
33             .load(drama.imageUrl)
34             .into(holder.binding.imageView)
35
36         holder.binding.buttonDetail.setOnClickListener {
37             onDetailClick(drama)
38         }
39     }
40 }

```

*Tabel 7. Source Code Jawaban Soal No. 1 Modul 3*

#### 4. HomeFragment.kt

```

1 package                                com.example.modul3
2
3 import                                android.os.Bundle
4 import                                android.view.LayoutInflater
5 import                                android.view.View
6 import                                android.view.ViewGroup
7 import                                androidx.fragment.app.Fragment
8 import                                androidx.recyclerview.widget.LinearLayoutManager
9 import                                com.example.modul3.databinding.FragmentHomeBinding
10

```

```

11 class HomeFragment : Fragment(R.layout.fragment_home) {
12     private lateinit var binding: FragmentHomeBinding
13     private lateinit var dramaAdapter: ListDramaAdapter
14
15     override fun onCreateView(
16         inflater: LayoutInflater, container: ViewGroup?,
17         savedInstanceState: Bundle?
18     ): View? {
19         binding = FragmentHomeBinding.inflate(inflater,
20 container, false)
21
22         val dramas = listOf(
23             Drama(
24                 "Mysterious Lotus Casebook",
25                 "2023",
26                 "Mystery, Wuxia",
27                 "40 Episodes",
28                 "https://i.mydramalist.com/kAozjj_4c.jpg?v=1",
29                 "https://www.iq.com/album/mysterious-lotus-
30 casebook-2023-hg4cefqzed?lang=en_us"
31             ),
32             Drama(
33                 "Love Game In Eastern Fantasy",
34                 "2024",
35                 "Romance, Wuxia, Fantasy",
36                 "32 Episodes",
37                 "https://i.mydramalist.com/VXOLzy_4c.jpg?v=1",
38                 "https://wetv.vip/en/play/227hqhmxbvabwggz/d4100tnphtz"
39             ),
40             Drama(
41                 "Melody of Golden Age",
42                 "2024",
43                 "Historical, Mystery, Romance, Drama",
44                 "40 Episodes",
45                 "https://i.mydramalist.com/d0Q62d_4c.jpg?v=1",
46                 "https://www.iq.com/album/melody-of-golden-age-
47 2025-vobwm47vvd?lang=en_us"
48             ),
49             Drama(
50                 "One And Only",
51                 "2021",
52                 "Military, Historical, Romance, Political",
53                 "24 Episodes",
54                 "https://i.mydramalist.com/BLrkR_4c.jpg?v=1",
55                 "https://www.iq.com/album/one-and-only-2021-
56 24mppdujjp9?lang=en_us"
57             ),
58             Drama(
59                 "Go Ahead",
60                 "2020",
61                 "Comedy, Romance, Drama, Melodrama",
62

```

```

63         "46                               Episodes",
64         "https://i.mydramalist.com/exXvQ_4c.jpg?v=1",
65         "https://www.iq.com/album/go-ahead-2020-
66 1bm24k36pk9?lang=en_us"
67     )
68 )
69
70     dramaAdapter = ListDramaAdapter(dramas) { selectedDrama
71 ->
72         val         fragment         =         DetailFragment()
73         val         bundle         =         Bundle()
74         bundle.putParcelable("drama",         selectedDrama)
75         fragment.arguments         =         bundle
76
77         parentFragmentManager.beginTransaction()
78             .replace(R.id.fragment_container,         fragment)
79             .addToBackStack(null)
80             .commit()
81     }
82
83     binding.recyclerView.apply                                     {
84         layoutManager         =         LinearLayoutManager(context)
85         adapter         =         dramaAdapter
86     }
87     return                                     binding.root
88 }
89 }

```

*Tabel 8. Source Code Jawaban Soal No. 1 Modul 3*

## 5. DetailFragment.kt

```

1  package                                     com.example.modul3
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.modul3.databinding.FragmentDetailBinding
11
12 class                                     DetailFragment         :         Fragment()         {
13     private lateinit var binding:         FragmentDetailBinding
14
15     override fun onCreateView(
16         inflater:         LayoutInflater,         container:         ViewGroup?,
17         savedInstanceState:         Bundle?
18     ):         View?         {
19         binding         =         FragmentDetailBinding.inflate(inflater,
20 container,         false)
21

```

```

22         val drama = arguments?.getParcelable<Drama>("drama")
23         drama?.let { selectedDrama ->
24             binding.drama = selectedDrama
25
26             binding.openLinkButton.setOnClickListener {
27                 val intent = Intent(Intent.ACTION_VIEW,
28 Uri.parse(selectedDrama.link))
29                 startActivity(intent)
30             }
31
32             binding.backButton.setOnClickListener {
33
34 requireActivity().supportFragmentManager.popBackStack()
35             }
36         }
37
38         return binding.root
39     }
40 }

```

*Tabel 9. Source Code Jawaban Soal No. 1 Modul 3*

## 6. activity\_main.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent">
8
9      <FrameLayout
10         android:id="@+id/fragment_container"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent" />
13 </androidx.constraintlayout.widget.ConstraintLayout>

```

*Tabel 10. Source Code Jawaban Soal No. 1 Modul 3*

## 7. item\_drama.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <layout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto">
5
6      <data>
7          <variable
8              name="drama"
9              type="com.example.modul3.Drama" />
10         </data>
11

```



```

12     <androidx.cardview.widget.CardView
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:layout_margin="8dp"
16         android:padding="8dp"
17         android:elevation="4dp"
18         android:clickable="true"
19         android:focusable="true">
20
21     <androidx.constraintlayout.widget.ConstraintLayout
22         android:layout_width="match_parent"
23         android:layout_height="wrap_content">
24
25         <ImageView
26             android:id="@+id/imageView"
27             android:layout_width="0dp"
28             android:layout_height="0dp"
29             android:layout_marginEnd="8dp"
30
31             android:contentDescription="@string/detail_image"
32             android:scaleType="centerCrop"
33             app:layout_constraintEnd_toEndOf="parent"
34             app:layout_constraintHeight_default="percent"
35             app:layout_constraintHeight_percent="0.5"
36             app:layout_constraintHorizontal_bias="0.0"
37             app:layout_constraintStart_toStartOf="parent"
38             app:layout_constraintTop_toTopOf="parent"      />
39
40         <TextView
41             android:id="@+id/titleText"
42             android:layout_width="wrap_content"
43             android:layout_height="wrap_content"
44             android:text="@{drama.title}"
45             android:textSize="16sp"
46             app:layout_constraintEnd_toEndOf="parent"
47             app:layout_constraintStart_toStartOf="parent"
48
49             app:layout_constraintTop_toBottomOf="@id/imageView"      />
50
51         <TextView
52             android:id="@+id/yearText"
53             android:layout_width="wrap_content"
54             android:layout_height="wrap_content"
55             android:text="@{drama.year}"
56             android:textSize="14sp"
57             app:layout_constraintEnd_toEndOf="parent"
58             app:layout_constraintStart_toStartOf="parent"
59
60             app:layout_constraintTop_toBottomOf="@id/titleText"      />
61
62         <Button
63             android:id="@+id/buttonDetail"

```

64	android:layout_width="wrap_content"
65	android:layout_height="wrap_content"
66	android:text="Detail"
67	android:layout_marginTop="8dp"
68	
69	app:layout_constraintTop_toBottomOf="@id/yearText"
70	app:layout_constraintStart_toStartOf="parent"
71	app:layout_constraintEnd_toEndOf="parent"/>
72	
73	</androidx.constraintlayout.widget.ConstraintLayout>
74	</androidx.cardview.widget.CardView>
75	</layout>

Tabel 11. Source Code Jawaban Soal No. 1 Modul 3

## 8. fragment\_home.xml

1	<?xml	version="1.0"	encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout		
3	xmlns:android="http://schemas.android.com/apk/res/android"		
4	android:layout_width="match_parent"		
5	android:layout_height="match_parent"		
6	xmlns:app="http://schemas.android.com/apk/res-auto">		
7			
8	<androidx.recyclerview.widget.RecyclerView		
9	android:id="@+id/recyclerView"		
10	android:layout_width="0dp"		
11	android:layout_height="0dp"		
12	app:layout_constraintBottom_toBottomOf="parent"		
13	app:layout_constraintEnd_toEndOf="parent"		
14	app:layout_constraintStart_toStartOf="parent"		
15	app:layout_constraintTop_toTopOf="parent"		/>
16	</androidx.constraintlayout.widget.ConstraintLayout>		

Tabel 12. Source Code Jawaban Soal No. 1 Modul 3

## 9. fragment\_detail.xml

1	<?xml	version="1.0"	encoding="utf-8"?>
2	<layout		
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			
7	<data>		
8	<variable		
9	name="drama"		
10	type="com.example.modul3.Drama"		/>
11	</data>		
12			
13	<ScrollView		
14	android:layout_width="match_parent"		
15	android:layout_height="match_parent"		

```

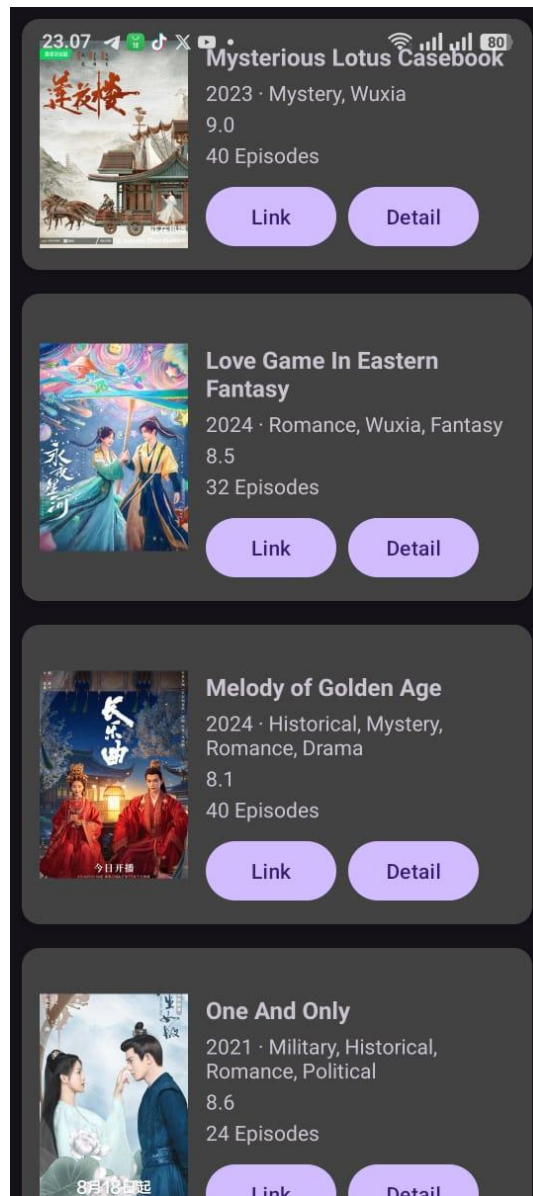
16         tools:context=".DetailFragment">
17
18         <LinearLayout
19             android:orientation="vertical"
20             android:padding="16dp"
21             android:layout_width="match_parent"
22             android:layout_height="wrap_content"
23             android:gravity="center_horizontal">
24
25             <ImageView
26                 android:id="@+id/detail_image"
27                 android:layout_width="match_parent"
28                 android:layout_height="500dp"
29                 android:scaleType="centerCrop"
30
31                 android:contentDescription="@string/detail_image"
32                 app:imageUrl="@{drama.imageUrl}" />
33
34             <TextView
35                 android:id="@+id/title_text"
36                 android:layout_width="wrap_content"
37                 android:layout_height="wrap_content"
38                 android:text="@{drama.title}"
39                 android:textSize="20sp"
40                 android:textStyle="bold"
41                 android:paddingTop="8dp"
42                 android:layout_gravity="center_horizontal"/>
43
44             <TextView
45                 android:id="@+id/year_text"
46                 android:layout_width="wrap_content"
47                 android:layout_height="wrap_content"
48                 android:text="@{drama.year}"
49                 android:textSize="16sp"
50                 android:paddingTop="4dp"
51                 android:layout_gravity="center_horizontal"/>
52
53             <TextView
54                 android:id="@+id/genre_text"
55                 android:layout_width="wrap_content"
56                 android:layout_height="wrap_content"
57                 android:text="@{drama.genre}"
58                 android:textSize="16sp"
59                 android:paddingTop="4dp"
60                 android:layout_gravity="center_horizontal"/>
61
62             <TextView
63                 android:id="@+id/episodes_text"
64                 android:layout_width="wrap_content"
65                 android:layout_height="wrap_content"
66                 android:text="@{drama.episodes}"
67                 android:textSize="16sp"

```

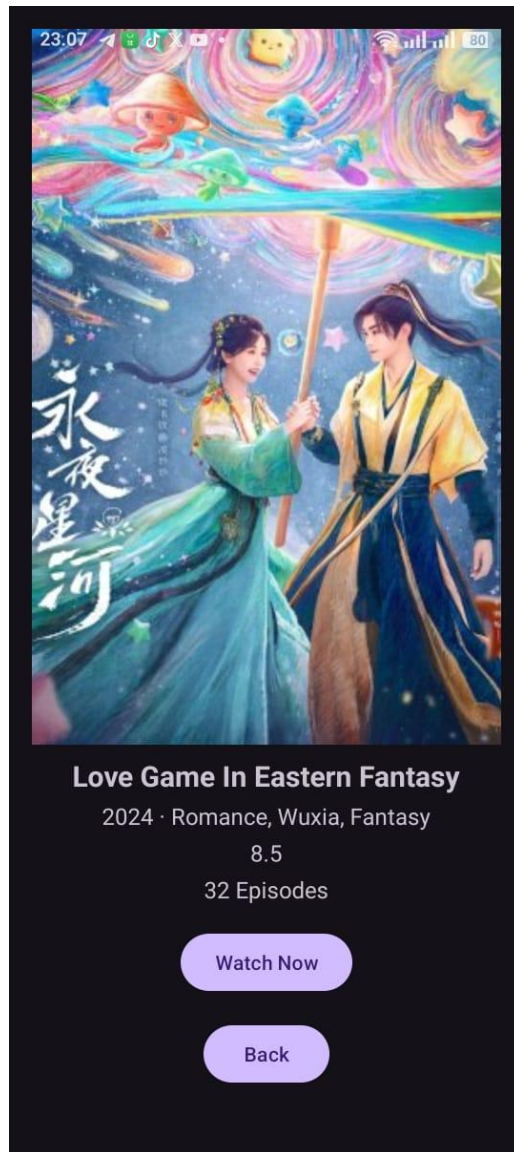
68	android:paddingTop="4dp"
69	android:layout_gravity="center_horizontal"/>
70	
71	<Button
72	android:id="@+id/openLinkButton"
73	android:layout_width="wrap_content"
74	android:layout_height="wrap_content"
75	android:text="Watch Now"
76	android:layout_marginTop="16dp"
77	android:layout_gravity="center_horizontal" />
78	
79	<Button
80	android:id="@+id/backButton"
81	android:layout_width="wrap_content"
82	android:layout_height="wrap_content"
83	android:text="Back"
84	android:layout_marginTop="16dp"
85	android:layout_gravity="center_horizontal"/>
86	</LinearLayout>
87	</ScrollView>
88	</layout>

*Tabel 13. Source Code Jawaban Soal No. 1 Modul 3*

## B. Output Program



Gambar 7. Screenshot Hasil Jawaban Soal 1



Gambar 8. Screenshot Hasil Jawaban Soal 1

### C. Pembahasan

- **MainActivity.kt:**

File ini merupakan titik awal aplikasi Android, tempat di mana layout utama ditentukan dan fragment pertama dimuat. Di dalam metode `onCreate()`, fungsi `setContentView()` digunakan untuk menetapkan layout XML utama (`activity_main.xml`) yang berisi `FrameLayout` sebagai wadah fragment. Selanjutnya, ada pemeriksaan terhadap `savedInstanceState`, yang digunakan untuk memastikan bahwa fragment `HomeFragment` hanya ditambahkan ketika aplikasi

pertama kali dijalankan, bukan saat dipulihkan dari keadaan sebelumnya. Jika kondisinya terpenuhi (yakni `savedInstanceState == null`), maka fragment `HomeFragment` dimuat ke dalam `fragment_container` menggunakan `supportFragmentManager`. Hal ini memastikan pengguna langsung melihat daftar drama saat aplikasi dibuka, dan fragment tidak dimuat ulang secara tidak perlu saat orientasi layar berubah atau aplikasi dikembalikan dari latar belakang.

- **Drama.kt**

File ini berisi sebuah data class bernama `Drama`, yang digunakan sebagai model data dalam aplikasi. Data class ini menyimpan lima properti utama: `title` (judul drama), `yearGenre` (kombinasi tahun rilis dan genre), `episodes` (jumlah episode), `imageUrl` (alamat URL gambar poster drama), dan `link` (URL video atau halaman nonton). Dengan adanya data class ini, setiap objek drama dapat dengan mudah dibuat dan dikelola di dalam daftar. Karena `Drama` mengimplementasikan `Parcelable`, objek-objek ini bisa dengan mudah dikirimkan antar fragment melalui `Bundle`, yang sangat penting dalam navigasi aplikasi.

- **ListDramaAdapter.kt**

File ini berperan sebagai jembatan antara data dan tampilan pada `RecyclerView`. Kelas `DramaAdapter` mengambil daftar objek `Drama` serta sebuah fungsi callback bernama `onDetailClick` yang akan dipanggil saat pengguna menekan tombol "Detail". Di dalamnya terdapat kelas `DramaViewHolder`, yang mengikat data ke layout `item_drama.xml` menggunakan data binding. Pada metode `onBindViewHolder`, adapter akan menetapkan data `Drama` ke properti `binding.drama` dan juga menetapkan listener tombol "Detail". Saat tombol ditekan, fungsi `onDetailClick` akan dijalankan dengan parameter objek `Drama` terkait, sehingga aktivitas atau fragment dapat merespons aksi pengguna untuk menampilkan detail. Struktur ini memisahkan logika tampilan dan logika interaksi, membuat kode lebih modular dan mudah dirawat.

- **HomeFragment.kt**

Fragment ini bertugas menampilkan daftar drama yang tersedia kepada pengguna. Pada saat `onCreateView()`, fragment ini menggunakan `FragmentHomeBinding` untuk menghubungkan

layout `fragment_home.xml` dengan kode Kotlin. `RecyclerView` yang ada di layout kemudian dihubungkan dengan `DramaAdapter`, dan daftar drama diisi menggunakan fungsi `getListDrama()` yang berisi data dummy. Fungsi ini menciptakan beberapa objek `Drama` secara manual dan menambahkannya ke dalam list. Salah satu bagian penting di sini adalah penanganan tombol "Detail" pada setiap item. Saat tombol tersebut ditekan, aplikasi akan membuat instance `DetailFragment`, mengirim data drama melalui `Bundle`, dan mengganti tampilan fragment saat ini dengan `DetailFragment`. Proses ini dilakukan menggunakan `parentFragmentManager` dan transaksi fragment, memungkinkan navigasi antar halaman tanpa perlu memulai aktivitas baru.

- **DetailFragment.kt**

Fragment ini digunakan untuk menampilkan informasi lengkap dari drama yang dipilih oleh pengguna. Di dalam metode `onCreateView()`, data `Drama` diterima dari arguments menggunakan metode `getParcelable()`, yang sebelumnya dikirim dari `HomeFragment`. Data ini kemudian diikat ke layout menggunakan `FragmentDetailBinding`, sehingga semua elemen UI seperti judul, genre, jumlah episode, dan gambar secara otomatis terisi dengan data yang sesuai. Fragment ini juga menyediakan dua aksi utama bagi pengguna. Pertama, tombol "Watch Now" yang menggunakan `Intent.ACTION_VIEW` untuk membuka link video di browser atau aplikasi terkait. Kedua, tombol "Back" yang memungkinkan pengguna kembali ke daftar drama dengan memanggil `requireActivity().onBackPressedDispatcher.onBackPressed()`, menggantikan fungsi `popBackStack()` agar tetap kompatibel dengan berbagai versi Android.

- **activity\_main.xml**

File ini adalah layout utama untuk aplikasi, yang menggunakan `ConstraintLayout` sebagai root view. Layout ini hanya memiliki satu elemen, yaitu `FrameLayout` dengan ID `fragment_container`, yang akan menjadi wadah untuk menampilkan fragment lainnya. `FrameLayout` ini mengisi seluruh layar dan akan digunakan untuk menampung fragment yang akan di-embed dalam activity ini. Jadi, layout utama aplikasi hanya berfokus pada tempat untuk memuat fragment, tanpa ada elemen tampilan lainnya.



- **item\_drama.xml**

File ini mendesain tampilan item dalam daftar drama yang akan ditampilkan di dalam RecyclerView. Pada bagian awal, dideklarasikan sebuah variabel drama yang bertipe `com.example.modul3.Drama` menggunakan data binding. Di dalam layout, terdapat sebuah `CardView` dengan margin, padding, dan elevasi untuk memberikan efek bayangan. Di dalam `CardView`, terdapat beberapa elemen tampilan: sebuah `ImageView` untuk menampilkan gambar drama dengan proporsi yang diatur agar terlihat pas, dan dua `TextView` untuk menampilkan judul dan tahun dari drama. Selain itu, ada sebuah tombol "Detail" yang digunakan untuk menunjukkan detail dari drama ketika diklik. Semua elemen ini diletakkan dengan bantuan `ConstraintLayout`, yang memastikan elemen-elemen tersebut terorganisir dengan baik di dalam layout.

- **fragment\_home.xml**

File ini adalah layout untuk fragment yang menampilkan daftar drama dalam bentuk RecyclerView. RecyclerView diatur untuk mengisi seluruh ukuran fragment, dari atas ke bawah dan kiri ke kanan, berkat penggunaan `ConstraintLayout` dan pengaturan constraint yang tepat. Fragment ini bertugas untuk menampilkan daftar item dalam aplikasi, di mana setiap item tersebut akan menggunakan layout `item_drama.xml` yang telah didefinisikan sebelumnya. Dengan demikian, RecyclerView akan menjadi tempat di mana pengguna dapat menggulir dan melihat berbagai drama yang tersedia di aplikasi.

- **fragment\_detail.xml**

File ini digunakan untuk menampilkan detail dari sebuah drama yang dipilih. Layout menggunakan `ScrollView` untuk memastikan bahwa konten dapat digulir jika ukurannya melebihi layar. Di dalam `ScrollView`, terdapat `LinearLayout` dengan orientasi vertikal untuk mengatur elemen-elemen yang ada. Elemen pertama adalah `ImageView` yang menampilkan gambar drama, yang dihubungkan dengan URL gambar melalui data binding. Berikutnya ada beberapa `TextView` yang menampilkan informasi tentang drama, seperti judul, tahun, genre, dan jumlah episode, semuanya diatur agar terpusat secara horizontal dan diberi padding agar lebih rapi. Di bagian bawah, terdapat dua tombol: satu untuk membuka tautan ke video drama (dengan teks "Watch Now"), dan satu lagi untuk kembali ke halaman sebelumnya (dengan

teks "Back"). Semua elemen ini diatur dengan cara yang membuat tampilan detail drama terlihat terstruktur dan mudah dinavigasi.

## SOAL 2

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

### A. Penjelasan

RecyclerView masih banyak digunakan karena memberikan kontrol dan fleksibilitas tinggi untuk menampilkan data kompleks. Komponen ini mendukung berbagai fitur seperti animasi, penggunaan LayoutManager yang dapat disesuaikan, serta efisiensi memori melalui penggunaan ViewHolder. Meskipun lebih rumit dan memerlukan kode tambahan seperti adapter dan view holder, RecyclerView cocok untuk proyek yang berbasis View XML dan memerlukan kustomisasi tingkat lanjut. Sementara itu, LazyColumn dari Jetpack Compose menawarkan cara yang lebih sederhana dan deklaratif untuk menampilkan daftar. Penulisan kode lebih ringkas tanpa perlu membuat adapter, sehingga cocok untuk pengembangan modern berbasis Compose. Namun, LazyColumn memiliki keterbatasan dalam hal fleksibilitas dan kustomisasi jika dibandingkan dengan RecyclerView. Pilihan antara keduanya bergantung pada pendekatan UI yang digunakan dalam proyek.

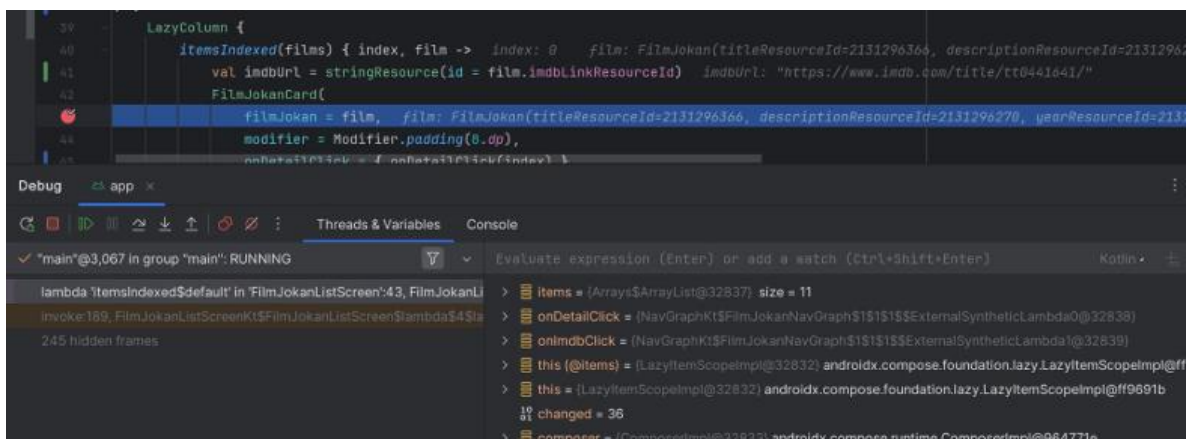
## MODUL 4: ViewModel and Debugging

### SOAL 1

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

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

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



Gambar 9. Soal No. 1 Modul 4

## A. Source Code

### 1. MainActivity.kt

```
1 package com.example.modul4
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import com.example.modul4.databinding.ActivityMainBinding
6
7 class MainActivity : AppCompatActivity() {
8
9     private lateinit var binding: ActivityMainBinding
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         binding = ActivityMainBinding.inflate(layoutInflater)
14         setContentView(binding.root)
15
16         if (savedInstanceState == null) {
17             supportFragmentManager.beginTransaction()
18                 .replace(R.id.fragment_container,
19 HomeFragment())
20                 .commit()
21         }
22     }
23 }
```

Tabel 14. Source Code Jawaban Soal No. 1 Modul 4

### 2. Drama.kt

```
1 package com.example.modul4
2
3 import android.os.Parcelable
4 import kotlinx.parcelize.Parcelize
5
6 @Parcelize
7 data class Drama (
8     val title: String,
9     val yearGenre: String,
10    val rating: String,
11    val episodes: String,
12    val imageUrl: String,
13    val link: String
14 ) : Parcelable
```

Tabel 15. Source Code Jawaban Soal No. 1 Modul 4

### 3. ListDramaAdapter.kt

```
1 package com.example.modul4
2
3 import android.content.Intent
4 import android.net.Uri
```

5	import	android.view.LayoutInflater
6	import	android.view.ViewGroup
7	import	androidx.recyclerview.widget.RecyclerView
8	import	com.bumptech.glide.Glide
9	import	com.example.modul4.databinding.ItemDramaBinding
10		
11	class	ListDramaAdapter(
12	private	val listDrama: List<Drama>,
13	private	val onDetailClick: (Drama) -> Unit
14	)	: RecyclerView.Adapter<ListDramaAdapter.MyViewHolder>() {
15		
16	inner class	MyViewHolder(val binding: ItemDramaBinding) :
17	RecyclerView.ViewHolder	(binding.root)
18		
19	override fun	onCreateViewHolder(parent: ViewGroup, viewType:
20	Int):	MyViewHolder {
21	val	binding =
22	ItemDramaBinding.inflate	(LayoutInflater.from(parent.context),
23	parent,	false)
24	return	MyViewHolder(binding)
25	}	
26		
27	override fun	getItemCount(): Int = listDrama.size
28		
29	override fun	onBindViewHolder(holder: MyViewHolder,
30	position:	Int) {
31	val	drama = listDrama[position]
32	val	context = holder.binding.root.context
33		
34	holder.binding.drama	= drama
35		
36	holder.binding.buttonDetail.setOnClickListener	{
37	onDetailClick(drama)	
38	}	
39		
40	holder.binding.buttonLink.setOnClickListener	{
41	val intent	= Intent(Intent.ACTION_VIEW,
42	Uri.parse(drama.link))	
43	context.startActivity(intent)	
44	}	
45	}	
46	}	

Tabel 16. Source Code Jawaban Soal No. 1 Modul 4

#### 4. HomeFragment.kt

1	package	com.example.modul4
2		
3	import	android.os.Bundle
4	import	android.util.Log
5	import	android.view.LayoutInflater
6	import	android.view.View

```

7 import                                android.view.ViewGroup
8 import                                androidx.fragment.app.Fragment
9 import                                androidx.fragment.app.viewModels
10 import                               androidx.lifecycle.lifecycleScope
11 import                                androidx.recyclerview.widget.LinearLayoutManager
12 import                                com.example.modul4.databinding.FragmentHomeBinding
13 import                                kotlin.coroutines.flow.collectLatest
14 import                                kotlin.coroutines.launch
15
16 class                                HomeFragment                                :                                Fragment()                                {
17
18     private    lateinit    var    binding:    FragmentHomeBinding
19     private    val    viewModel:    DramaViewModel    by    viewModels {
20         DramaViewModelFactory()
21     }
22     private    lateinit    var    dramaAdapter:    ListDramaAdapter
23
24     override                                fun                                onCreateView(
25         inflater:    LayoutInflater,    container:    ViewGroup?,
26         savedInstanceState:    Bundle?
27     ):    View                                {
28         binding    =    FragmentHomeBinding.inflate(inflater,
29         container,
30         return                                binding.root
31     }
32
33     override fun onCreateView(view: View, savedInstanceState:
34 Bundle?)                                {
35         dramaAdapter = ListDramaAdapter(emptyList()) { drama ->
36             Log.d("HomeFragment",    "Detail    clicked:
37 ${drama.title}")
38             val    fragment    =    DetailFragment().apply    {
39                 arguments    =    Bundle().apply    {
40                     putParcelable("drama",    drama)
41                 }
42             }
43             parentFragmentManager.beginTransaction()
44                 .replace(R.id.fragment_container,    fragment)
45                 .addToBackStack(null)
46                 .commit()
47         }
48
49         binding.recyclerView.apply                                {
50             layoutManager    =    LinearLayoutManager(context)
51             adapter    =    dramaAdapter
52         }
53
54         viewLifecycleOwner.lifecycleScope.launch                                {
55             viewModel.dramas.collectLatest    {    list    ->
56                 Log.d("HomeFragment",    "List    updated:
57 ${list.size}    items")
58             dramaAdapter = ListDramaAdapter(list) { drama -

```

```

59 >
60         Log.d("HomeFragment", "Detail clicked:
61 ${drama.title}")
62         val fragment = DetailFragment().apply {
63             arguments = Bundle().apply {
64                 putParcelable("drama", drama)
65             }
66         }
67         parentFragmentManager.beginTransaction()
68             .replace(R.id.fragment_container,
69 fragment)
70             .addToBackStack(null)
71             .commit()
72     }
73     binding.recyclerView.adapter = dramaAdapter
74 }
75 }
76 }
77 }

```

Tabel 17. Source Code Jawaban Soal No. 1 Modul 4

## 5. DetailFragment.kt

```

1 package com.example.modul4
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.modul4.databinding.FragmentDetailBinding
11
12 class DetailFragment : Fragment() {
13     private lateinit var binding: FragmentDetailBinding
14
15     override fun onCreateView(
16         inflater: LayoutInflater, container: ViewGroup?,
17         savedInstanceState: Bundle?
18     ): View? {
19         binding = FragmentDetailBinding.inflate(inflater,
20 container, false)
21
22         val drama = arguments?.getParcelable<Drama>("drama")
23         drama?.let { selectedDrama ->
24             binding.drama = selectedDrama
25
26             binding.openLinkButton.setOnClickListener {
27                 val intent = Intent(Intent.ACTION_VIEW,
28 Uri.parse(selectedDrama.link))
29                 startActivity(intent)

```



```

30         }
31
32         binding.backButton.setOnClickListener {
33
34             requireActivity().supportFragmentManager.popBackStack()
35         }
36     }
37
38     return binding.root
39 }
40 }

```

Tabel 18. Source Code Jawaban Soal No. 1 Modul 4

## 6. DramaViewModel.kt

```

1 package com.example.modul4
2
3 import android.util.Log
4 import androidx.lifecycle.ViewModel
5 import kotlinx.coroutines.flow.MutableStateFlow
6 import kotlinx.coroutines.flow.StateFlow
7
8 class DramaViewModel : ViewModel() {
9
10     private val _dramas =
11     MutableStateFlow<List<Drama>>(emptyList())
12     val dramas: StateFlow<List<Drama>> = _dramas
13
14     private val _selectedDrama = MutableStateFlow<Drama?>(null)
15     val selectedDrama: StateFlow<Drama?> = _selectedDrama
16
17     init {
18         _dramas.value = listOf(
19             Drama(
20                 "Mysterious Lotus Casebook", "2023 · Mystery,
21 Wuxia", "Rating: 8.7", "40 Episodes",
22                 "https://i.mydramalist.com/kAozjj_4c.jpg?v=1",
23                 "https://www.iq.com/album/mysterious-lotus-
24 casebook-2023-hg4cefqzed?lang=en_us"
25             ),
26             Drama(
27                 "Love Game In Eastern Fantasy", "2024 · Romance,
28 Wuxia, Fantasy", "Rating: 8.5", "32 Episodes",
29                 "https://i.mydramalist.com/VXOLzy_4c.jpg?v=1",
30                 "https://wetv.vip/en/play/227hqhmxxvabwggz/d4100tnphtz"
31             ),
32             Drama(
33                 "Melody of Golden Age", "2024 · Historical,
34 Mystery, Romance, Drama", "Rating: 8.1", "40 Episodes",
35                 "https://i.mydramalist.com/d0Q62d_4c.jpg?v=1",
36                 "https://www.iq.com/album/melody-of-golden-age-
37

```

```

38 2025-vobwm47vvd?lang=en_us"
39     ),
40     Drama (
41         "One And Only", "2021 · Military, Historical,
42 Romance, Political", "Rating: 8.6", "24 Episodes",
43         "https://i.mydramalist.com/BLrkR_4c.jpg?v=1",
44         "https://www.iq.com/album/one-and-only-2021-
45 24mppdujjp9?lang=en_us"
46     ),
47     Drama (
48         "The Prisoner of Beauty", "2025 · Historical,
49 Romance, Political", "Rating: 8.9", "36 Episodes",
50         "https://i.mydramalist.com/mOE1XJ_4c.jpg?v=1",
51
52 "https://wetv.vip/en/play/nwo9p9nml6dgm8l/e41010dzbwf-
53 EP01%3A_The_Prisoner_of_Beauty"
54     )
55 )
56
57 Log.d("DramaViewModel", "Drama list initialized with
58 ${_dramas.value.size} items")
59 }
60
61 fun selectDrama(drama: Drama) {
62     _selectedDrama.value = drama
63     Log.d("DramaViewModel", "Drama selected:
64 ${drama.title}")
65 }
66 }

```

Tabel 19. Source Code Jawaban Soal No. 1 Modul 4

## 7. DramaViewModelFactory.kt

```

1 package com.example.modul4
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5
6 class DramaViewModelFactory : ViewModelProvider.Factory {
7     override fun <T : ViewModel> create(modelClass: Class<T>):
8     T {
9         if
10 (modelClass.isAssignableFrom(DramaViewModel::class.java)) {
11             return DramaViewModel() as T
12         }
13         throw IllegalArgumentException("Unknown ViewModel
14 class")
15     }
16 }

```

Tabel 20. Source Code Jawaban Soal No. 1 Modul 4

## 8. activity\_main.xml

```
1  <?xml                      version="1.0"                      encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent">
8
9      <FrameLayout
10         android:id="@+id/fragment_container"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"                                />
13 </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 21. Source Code Jawaban Soal No. 1 Modul 4

## 9. item\_drama.xml

```
1  <?xml                      version="1.0"                      encoding="utf-8"?>
2  <layout
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
7      <data>
8          <variable
9              name="drama"
10             type="com.example.modul4.Drama"                                />
11      </data>
12
13      <androidx.cardview.widget.CardView
14          android:layout_width="match_parent"
15          android:layout_height="wrap_content"
16          android:layout_margin="8dp"
17          app:cardCornerRadius="12dp"
18          app:cardElevation="4dp">
19
20          <androidx.constraintlayout.widget.ConstraintLayout
21              android:layout_width="match_parent"
22              android:layout_height="wrap_content"
23              android:padding="12dp">
24
25              <ImageView
26                  android:id="@+id/imageView"
27                  android:layout_width="100dp"
28                  android:layout_height="140dp"
29
30                  android:contentDescription="@string/detail_image"
31                  android:scaleType="centerCrop"
32                  app:imageUrl="@{drama.imageUrl}"
33              />
26          />
18      />
```

```

34 app:layout_constraintBottom_toBottomOf="parent"
35         app:layout_constraintStart_toStartOf="parent"
36         app:layout_constraintTop_toTopOf="parent"
37         tools:src="@tools:sample/avatars"      />
38
39     <TextView
40         android:id="@+id/titleText"
41         android:layout_width="0dp"
42         android:layout_height="wrap_content"
43         android:layout_marginStart="12dp"
44         android:text="@{drama.title}"
45         android:textSize="16sp"
46         android:textStyle="bold"
47         app:layout_constraintEnd_toEndOf="parent"
48
49 app:layout_constraintStart_toEndOf="@id/imageView"
50
51 app:layout_constraintTop_toTopOf="@id/imageView"      />
52
53     <TextView
54         android:id="@+id/yearText"
55         android:layout_width="0dp"
56         android:layout_height="wrap_content"
57         android:layout_marginTop="4dp"
58         android:text="@{drama.yearGenre}"
59         android:textSize="14sp"
60         app:layout_constraintEnd_toEndOf="parent"
61
62 app:layout_constraintStart_toStartOf="@+id/titleText"
63
64 app:layout_constraintTop_toBottomOf="@id/titleText"      />
65
66     <TextView
67         android:id="@+id/info1Text"
68         android:layout_width="0dp"
69         android:layout_height="wrap_content"
70         android:layout_marginTop="2dp"
71         android:text="@{drama.rating}"
72         app:layout_constraintEnd_toEndOf="parent"
73
74 app:layout_constraintStart_toStartOf="@+id/titleText"
75
76 app:layout_constraintTop_toBottomOf="@id/yearText"      />
77
78     <TextView
79         android:id="@+id/info2Text"
80         android:layout_width="0dp"
81         android:layout_height="wrap_content"
82         android:layout_marginTop="2dp"
83         android:text="@{drama.episodes}"
84         app:layout_constraintEnd_toEndOf="parent"
85

```

86	app:layout_constraintStart_toStartOf="@+id/titleText"	
87		
88	app:layout_constraintTop_toBottomOf="@id/info1Text"	/>
89		
90	<Button	
91	android:id="@+id/buttonLink"	
92	android:layout_width="wrap_content"	
93	android:layout_height="wrap_content"	
94	android:layout_marginTop="8dp"	
95	android:text="Link"	
96		
97	app:layout_constraintStart_toStartOf="@+id/titleText"	
98		
99	app:layout_constraintTop_toBottomOf="@id/info2Text"	/>
100		
101	<Button	
102	android:id="@+id/buttonDetail"	
103	android:layout_width="wrap_content"	
104	android:layout_height="wrap_content"	
105	android:layout_marginStart="8dp"	
106	android:text="Detail"	
107		
108	app:layout_constraintStart_toEndOf="@id/buttonLink"	
109		
110	app:layout_constraintTop_toTopOf="@id/buttonLink"	/>
111		
112	</androidx.constraintlayout.widget.ConstraintLayout>	
113	</androidx.cardview.widget.CardView>	
114	</layout>	

Tabel 22. Source Code Jawaban Soal No. 1 Modul 4

## 10. fragment\_home.xml

1	<?xml	version="1.0"	encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout		
3	xmlns:android="http://schemas.android.com/apk/res/android"		
4	android:layout_width="match_parent"		
5	android:layout_height="match_parent"		
6	xmlns:app="http://schemas.android.com/apk/res-auto">		
7			
8	<androidx.recyclerview.widget.RecyclerView		
9	android:id="@+id/recyclerView"		
10	android:layout_width="0dp"		
11	android:layout_height="0dp"		
12	app:layout_constraintBottom_toBottomOf="parent"		
13	app:layout_constraintEnd_toEndOf="parent"		
14	app:layout_constraintStart_toStartOf="parent"		
15	app:layout_constraintTop_toTopOf="parent"		/>
16	</androidx.constraintlayout.widget.ConstraintLayout>		

Tabel 23. Source Code Jawaban Soal No. 1 Modul 4

## 11. fragment\_detail.xml

```
1  <?xml                      version="1.0"                      encoding="utf-8"?>
2  <layout
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
7      <data>
8          <variable
9              name="drama"
10             type="com.example.modul4.Drama"                      />
11      </data>
12
13      <ScrollView
14          android:layout_width="match_parent"
15          android:layout_height="match_parent"
16          tools:context=".DetailFragment">
17
18          <LinearLayout
19              android:orientation="vertical"
20              android:padding="16dp"
21              android:layout_width="match_parent"
22              android:layout_height="wrap_content"
23              android:gravity="center_horizontal">
24
25              <ImageView
26                  android:id="@+id/detail_image"
27                  android:layout_width="match_parent"
28                  android:layout_height="500dp"
29                  android:scaleType="centerCrop"
30
31                  android:contentDescription="@string/detail_image"
32                  app:imageUrl="@{drama.imageUrl}"                      />
33
34              <TextView
35                  android:id="@+id/title_text"
36                  android:layout_width="wrap_content"
37                  android:layout_height="wrap_content"
38                  android:text="@{drama.title}"
39                  android:textSize="20sp"
40                  android:textStyle="bold"
41                  android:paddingTop="8dp"
42                  android:layout_gravity="center_horizontal"/>
43
44              <TextView
45                  android:id="@+id/year_genre_text"
46                  android:layout_width="wrap_content"
47                  android:layout_height="wrap_content"
48                  android:text="@{drama.yearGenre}"
49                  android:textSize="16sp"
50                  android:paddingTop="4dp"
```

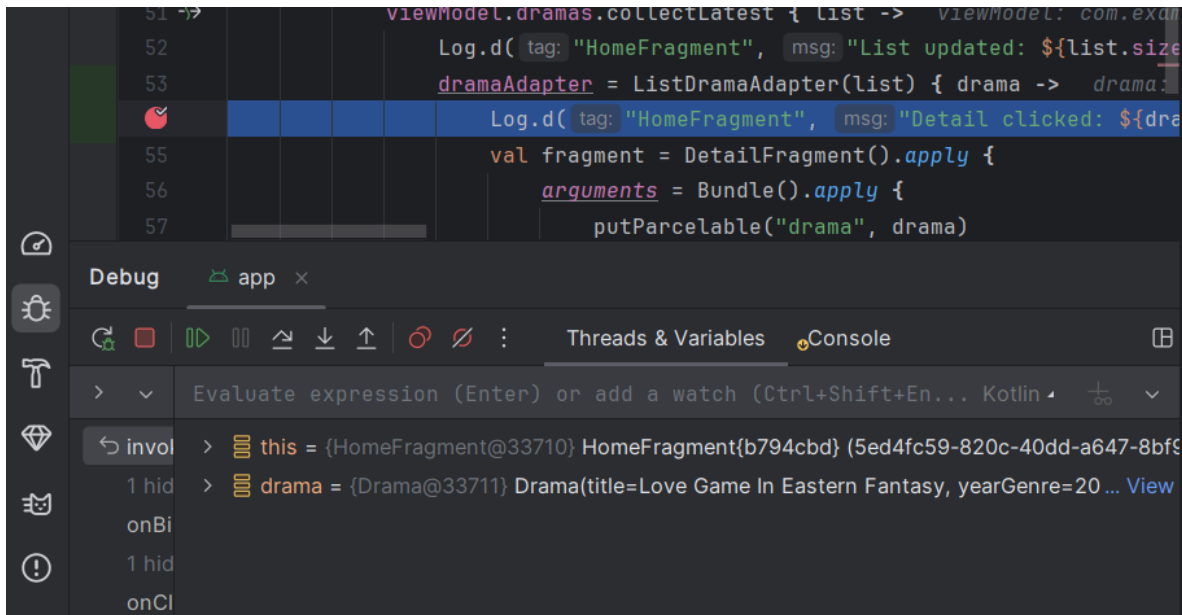
```

51         android:layout_gravity="center_horizontal"/>
52
53     <TextView
54         android:id="@+id/rating_text"
55         android:layout_width="wrap_content"
56         android:layout_height="wrap_content"
57         android:text="@{drama.rating}"
58         android:textSize="16sp"
59         android:paddingTop="4dp"
60         android:layout_gravity="center_horizontal"/>
61
62     <TextView
63         android:id="@+id/episodes_text"
64         android:layout_width="wrap_content"
65         android:layout_height="wrap_content"
66         android:text="@{drama.episodes}"
67         android:textSize="16sp"
68         android:paddingTop="4dp"
69         android:layout_gravity="center_horizontal"/>
70
71     <Button
72         android:id="@+id/openLinkButton"
73         android:layout_width="wrap_content"
74         android:layout_height="wrap_content"
75         android:text="@string/watch_here"
76         android:layout_marginTop="16dp"
77         android:layout_gravity="center_horizontal"    />
78
79     <Button
80         android:id="@+id/backButton"
81         android:layout_width="wrap_content"
82         android:layout_height="wrap_content"
83         android:text="@string/button_back"
84         android:layout_marginTop="16dp"
85         android:layout_gravity="center_horizontal"/>
86     </LinearLayout>
87 </ScrollView>
88 </layout>

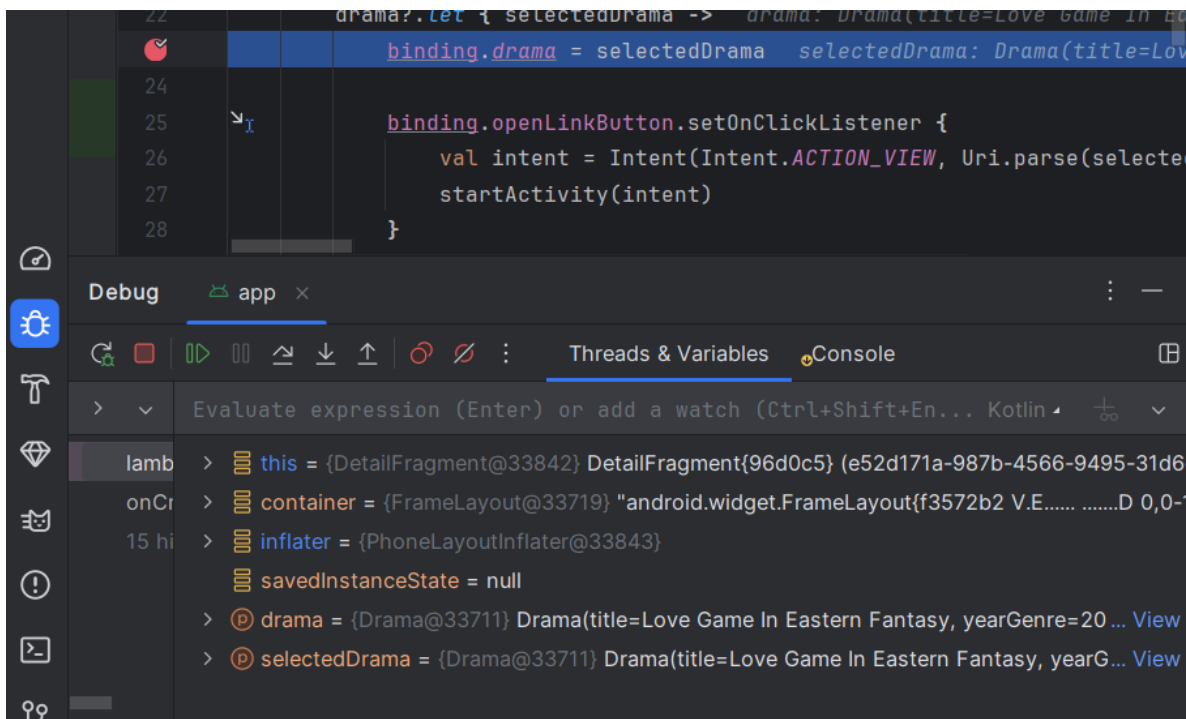
```

*Tabel 24. Source Code Jawaban Soal No. 1 Modul 4*

## B. Output Program

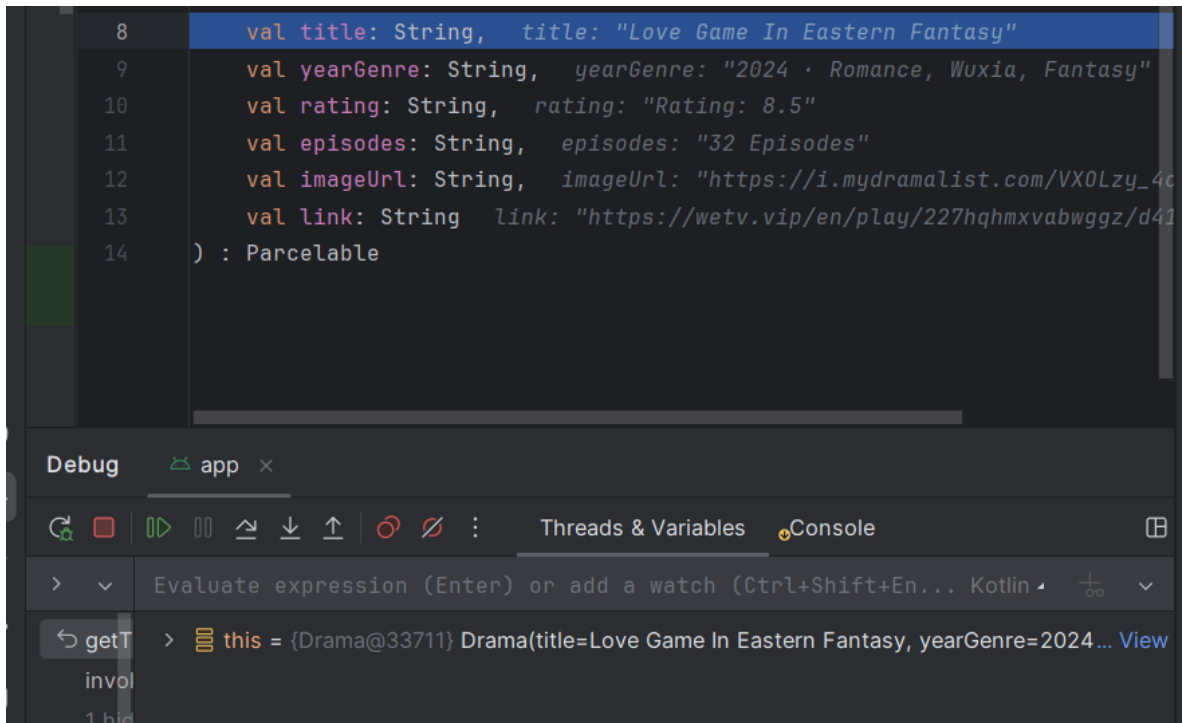


Gambar 10. Screenshot Hasil Jawaban Soal No. 1 Modul 4



Gambar 11. Screenshot Hasil Jawaban Soal No. 1 Modul 4





Gambar 12. Screenshot Hasil Jawaban Soal No. 1 Modul 4

### C. Pembahasan

- **MainActivity.kt:**

File ini merupakan titik awal aplikasi Android, tempat di mana layout utama ditentukan dan fragment pertama dimuat. Di dalam metode `onCreate()`, fungsi `setContentView()` digunakan untuk menetapkan layout XML utama (`activity_main.xml`) yang berisi `FrameLayout` sebagai wadah fragment. Selanjutnya, ada pemeriksaan terhadap `savedInstanceState`, yang digunakan untuk memastikan bahwa fragment `HomeFragment` hanya ditambahkan ketika aplikasi pertama kali dijalankan, bukan saat dipulihkan dari keadaan sebelumnya. Jika kondisinya terpenuhi (yakni `savedInstanceState == null`), maka fragment `HomeFragment` dimuat ke dalam `fragment_container` menggunakan `supportFragmentManager`. Hal ini memastikan pengguna langsung melihat daftar drama saat aplikasi dibuka, dan fragment tidak dimuat ulang secara tidak perlu saat orientasi layar berubah atau aplikasi dikembalikan dari latar belakang.

- **Drama.kt**

File ini berisi sebuah data class bernama Drama, yang digunakan sebagai model data dalam aplikasi. Data class ini menyimpan lima properti utama: title (judul drama), yearGenre (kombinasi tahun rilis dan genre), episodes (jumlah episode), imageUrl (alamat URL gambar poster drama), dan link (URL video atau halaman nonton). Dengan adanya data class ini, setiap objek drama dapat dengan mudah dibuat dan dikelola di dalam daftar. Karena Drama mengimplementasikan Parcelable, objek-objek ini bisa dengan mudah dikirimkan antar fragment melalui Bundle, yang sangat penting dalam navigasi aplikasi.

- **ListDramaAdapter.kt**

File ini berperan sebagai jembatan antara data dan tampilan pada RecyclerView. Kelas DramaAdapter mengambil daftar objek Drama serta sebuah fungsi callback bernama onDetailClick yang akan dipanggil saat pengguna menekan tombol "Detail". Di dalamnya terdapat kelas DramaViewHolder, yang mengikat data ke layout item\_drama.xml menggunakan data binding. Pada metode onBindViewHolder, adapter akan menetapkan data Drama ke properti binding.drama dan juga menetapkan listener tombol "Detail". Saat tombol ditekan, fungsi onDetailClick akan dijalankan dengan parameter objek Drama terkait, sehingga aktivitas atau fragment dapat merespons aksi pengguna untuk menampilkan detail. Struktur ini memisahkan logika tampilan dan logika interaksi, membuat kode lebih modular dan mudah dirawat.

- **HomeFragment.kt**

Fragment ini bertugas menampilkan daftar drama yang tersedia kepada pengguna. Pada saat onCreateView(), fragment ini menggunakan FragmentHomeBinding untuk menghubungkan layout fragment\_home.xml dengan kode Kotlin. RecyclerView yang ada di layout kemudian dihubungkan dengan DramaAdapter, dan daftar drama diisi menggunakan fungsi getListDrama() yang berisi data dummy. Fungsi ini menciptakan beberapa objek Drama secara manual dan menambahkannya ke dalam list. Salah satu bagian penting di sini adalah penanganan tombol "Detail" pada setiap item. Saat tombol tersebut ditekan, aplikasi akan membuat instance DetailFragment, mengirim data drama melalui Bundle, dan mengganti

tampilan fragment saat ini dengan DetailFragment. Proses ini dilakukan menggunakan parentFragmentManager dan transaksi fragment, memungkinkan navigasi antar halaman tanpa perlu memulai aktivitas baru.

- **DetailFragment.kt**

DetailFragment merupakan bagian dari aplikasi Android yang berfungsi menampilkan informasi detail dari sebuah drama yang dipilih oleh pengguna. Kelas ini menggunakan View Binding melalui objek FragmentDetailBinding untuk menghubungkan layout XML dengan logika di Kotlin secara langsung dan aman dari kesalahan penulisan ID view. Pada metode onCreateView(), layout di-inflate menggunakan FragmentDetailBinding.inflate() dan view utama dikembalikan sebagai root view dari fragment. Data Drama yang dikirim dari fragment sebelumnya diambil dari arguments dengan key "drama", lalu dicek keberadaannya. Jika data tersebut tidak null, maka akan digunakan untuk mengisi tampilan melalui binding (binding.drama = selectedDrama). Tombol openLinkButton disiapkan untuk membuka link dari drama menggunakan Intent eksplisit yang menjalankan browser dengan URL yang berasal dari properti link pada objek drama. Sedangkan backButton diprogram untuk menavigasi kembali ke fragment sebelumnya dengan memanggil popBackStack() pada supportFragmentManager. Seluruh interaksi ini dirancang untuk memberikan pengalaman pengguna yang intuitif dan responsif saat melihat detail sebuah drama.

- **DramaViewModel.kt**

DramaViewModel merupakan kelas turunan dari ViewModel yang berfungsi sebagai pengelola data untuk antarmuka pengguna pada aplikasi daftar drama. Di dalamnya terdapat dua properti utama yang dibungkus dengan MutableStateFlow, yaitu \_dramas untuk menyimpan daftar drama dan \_selectedDrama untuk menyimpan drama yang sedang dipilih pengguna. Kedua properti ini diekspose ke UI menggunakan StateFlow, memungkinkan komponen UI mengamati perubahan data secara reaktif dan efisien sesuai dengan prinsip arsitektur MVVM. Pada blok init, daftar drama diinisialisasi secara statis menggunakan beberapa objek Drama yang berisi informasi seperti judul, genre, rating, jumlah episode, gambar, dan tautan menonton. Selain itu, terdapat fungsi selectDrama() yang digunakan untuk memperbarui nilai selectedDrama ketika pengguna memilih salah satu item, serta

mencatat aktivitas tersebut melalui Log. Dengan memisahkan logika penyimpanan dan pengolahan data dari UI, kelas ini membantu menjaga kebersihan kode dan membuat aplikasi lebih mudah dirawat.

- **DramaViewModelFactory.kt**

DramaViewModelFactory adalah kelas yang mengimplementasikan ViewModelProvider.Factory dan digunakan untuk membuat instance DramaViewModel secara eksplisit. Factory ini diperlukan karena ViewModelProvider tidak bisa langsung membuat ViewModel dengan konstruktor kustom tanpa bantuan kelas pembantu. Dalam implementasinya, method create() memeriksa apakah modelClass merupakan turunan dari DramaViewModel, dan jika ya, maka objek baru akan dibuat dan dikembalikan. Jika tidak cocok, factory akan melemparkan pengecualian IllegalArgumentException. Penggunaan factory ini mempermudah pengelolaan dependensi dan mendukung penerapan arsitektur yang bersih dan skalabel, terutama ketika ViewModel di masa depan membutuhkan parameter atau dependensi tambahan seperti repository atau context.

- **activity\_main.xml**

File ini adalah layout utama untuk aplikasi, yang menggunakan ConstraintLayout sebagai root view. Layout ini hanya memiliki satu elemen, yaitu FrameLayout dengan ID fragment\_container, yang akan menjadi wadah untuk menampilkan fragment lainnya. FrameLayout ini mengisi seluruh layar dan akan digunakan untuk menampung fragment yang akan di-embed dalam activity ini. Jadi, layout utama aplikasi hanya berfokus pada tempat untuk memuat fragment, tanpa ada elemen tampilan lainnya.

- **item\_drama.xml**

File ini mendesain tampilan item dalam daftar drama yang akan ditampilkan di dalam RecyclerView. Pada bagian awal, dideklarasikan sebuah variabel drama yang bertipe com.example.modul3.Drama menggunakan data binding. Di dalam layout, terdapat sebuah CardView dengan margin, padding, dan elevasi untuk memberikan efek bayangan. Di dalam CardView, terdapat beberapa elemen tampilan: sebuah ImageView untuk menampilkan gambar drama dengan proporsi yang diatur agar terlihat pas, dan dua TextView untuk

menampilkan judul dan tahun dari drama. Selain itu, ada sebuah tombol "Detail" yang digunakan untuk menunjukkan detail dari drama ketika diklik. Semua elemen ini diletakkan dengan bantuan ConstraintLayout, yang memastikan elemen-elemen tersebut terorganisir dengan baik di dalam layout.

- **fragment\_home.xml**

File ini adalah layout untuk fragment yang menampilkan daftar drama dalam bentuk RecyclerView. RecyclerView diatur untuk mengisi seluruh ukuran fragment, dari atas ke bawah dan kiri ke kanan, berkat penggunaan ConstraintLayout dan pengaturan constraint yang tepat. Fragment ini bertugas untuk menampilkan daftar item dalam aplikasi, di mana setiap item tersebut akan menggunakan layout item\_drama.xml yang telah didefinisikan sebelumnya. Dengan demikian, RecyclerView akan menjadi tempat di mana pengguna dapat menggulir dan melihat berbagai drama yang tersedia di aplikasi.

- **fragment\_detail.xml**

File ini digunakan untuk menampilkan detail dari sebuah drama yang dipilih. Layout menggunakan ScrollView untuk memastikan bahwa konten dapat digulir jika ukurannya melebihi layar. Di dalam ScrollView, terdapat LinearLayout dengan orientasi vertikal untuk mengatur elemen-elemen yang ada. Elemen pertama adalah ImageView yang menampilkan gambar drama, yang dihubungkan dengan URL gambar melalui data binding. Berikutnya ada beberapa TextView yang menampilkan informasi tentang drama, seperti judul, tahun, genre, dan jumlah episode, semuanya diatur agar terpusat secara horizontal dan diberi padding agar lebih rapi. Di bagian bawah, terdapat dua tombol: satu untuk membuka tautan ke video drama (dengan teks "Watch Now"), dan satu lagi untuk kembali ke halaman sebelumnya (dengan teks "Back"). Semua elemen ini diatur dengan cara yang membuat tampilan detail drama terlihat terstruktur dan mudah dinavigasi.

## **Fungsi Debugger**

Debugger digunakan untuk mencari dan memperbaiki kesalahan (bug) dalam kode dengan menjalankan aplikasi secara bertahap. Alat ini membantu melihat nilai variabel dan alur eksekusi program secara langsung saat aplikasi berjalan.

## **Cara Menggunakan Debugger**

1. Pasang breakpoint di baris kode yang ingin dianalisis.
2. Jalankan aplikasi dalam mode debug.
3. Saat aplikasi berhenti di breakpoint, kamu bisa melihat isi variabel, stack trace, dan alur program.
4. Gunakan fitur step untuk menelusuri kode lebih dalam.

## **Fitur Step:**

- **Step Into:** Masuk ke dalam fungsi yang dipanggil.
- **Step Over:** Melewati fungsi tanpa masuk ke dalamnya.
- **Step Out:** Keluar dari fungsi dan kembali ke pemanggilnya.

## SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

### A. Penjelasan

Application class dalam arsitektur aplikasi Android adalah kelas dasar yang merepresentasikan seluruh siklus hidup aplikasi. Kelas ini dibuat satu kali saat aplikasi dijalankan dan tetap aktif selama aplikasi berjalan. Fungsi utama Application class adalah sebagai tempat untuk menginisialisasi komponen atau data yang harus tersedia secara global dan bertahan sepanjang aplikasi, seperti konfigurasi library, pengaturan dependency injection, atau objek shared resources. Dengan menggunakan Application class, developer dapat mengelola state dan resource yang bersifat global tanpa perlu mengulang inisialisasi di setiap activity atau fragment. Ini membuat aplikasi lebih efisien dan terstruktur karena sumber daya penting hanya diinisialisasi sekali pada saat aplikasi mulai berjalan. Application class juga dapat dimanfaatkan untuk menangani event global seperti perubahan konfigurasi atau memonitor lifecycle aplikasi secara keseluruhan.

## MODUL 5: Connect to the Internet

### SOAL 1

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

- 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.
- Gunakan KotlinX Serialization sebagai library JSON.
- Gunakan library seperti Coil atau Glide untuk image loading.
- API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: [Getting Started](#)
- Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)
- Gunakan caching strategy pada Room.
- 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. AppDatabase.kt

```
1 package com.example.modul5
2
3 import android.content.Context
4 import androidx.room.Database
5 import androidx.room.Room
6 import androidx.room.RoomDatabase
7
8 @Database(entities = [FavoriteTvShow::class], version = 1,
9 exportSchema = false)
10 abstract class AppDatabase : RoomDatabase() {
11     abstract fun favoriteDao(): FavoriteDao
12
13     companion object {
```



14	@Volatile
15	private        var        INSTANCE:        AppDatabase?        =        null
16	
17	fun        getDatabase(context:        Context):        AppDatabase        {
18	return        INSTANCE        ?:        synchronized(this)        {
19	val        instance        =        Room.databaseBuilder(
20	context.applicationContext,
21	AppDatabase::class.java,
22	"favorite_database"
23	).build()
24	INSTANCE                =                instance
25	instance
26	}
27	}
28	}
29	}

*Tabel 25. Source Code Jawaban Soal No.1 Modul 5*

## 2. BindingAdapter.kt

1	package	com.example.modul5
2		
3	import	android.widget.ImageView
4	import	androidx.databinding.BindingAdapter
5	import	com.bumptech.glide.Glide
6		
7	@BindingAdapter("imageUrl")	
8	fun        loadImage(view:        ImageView,        url:        String?)        {	
9	if                (!url.isNullOrEmpty())	{
10	Glide.with(view.context)	
11	.load(url)	
12	.into(view)	
13	}	
14	}	

*Tabel 26. Source Code Jawaban Soal No.1 Modul 5*

## 3. Detailfragment.kt

1	package	com.example.modul5
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.fragment.app.viewModels
11	import	androidx.lifecycle.lifecycleScope
12	import	com.example.modul5.databinding.FragmentDetailBinding
13	import	kotlinx.coroutines.launch
14		

```

15 class DetailFragment : Fragment() {
16
17     private lateinit var binding: FragmentDetailBinding
18     private val favoriteViewModel: FavoriteViewModel by
19     viewModels()
20
21     private var currentTvShow: TvShow? = null
22
23     override fun onCreateView(inflater: LayoutInflater,
24 container: ViewGroup?, savedInstanceState: Bundle?): View {
25         binding = FragmentDetailBinding.inflate(inflater,
26 container, false)
27         binding.lifecycleOwner = viewLifecycleOwner
28
29         val tvShow = arguments?.getParcelable<TvShow>("tvshow")
30         tvShow?.let {
31             currentTvShow = it
32             binding.tvShow = it
33             binding.executePendingBindings()
34
35             binding.openLinkButton.setOnClickListener {
36                 currentTvShow?.let { tv ->
37                     val intent = Intent(Intent.ACTION_VIEW,
38 Uri.parse("https://www.themoviedb.org/tv/${tv.id}"))
39                     startActivity(intent)
40                 }
41             }
42
43             binding.backButton.setOnClickListener {
44
45 requireActivity().supportFragmentManager.popBackStack()
46             }
47
48             lifecycleScope.launch {
49                 val isFav = favoriteViewModel.isFavorite(it)
50                 updateFavIcon(isFav)
51
52                 binding.favIconButton.setOnClickListener {
53                     favoriteViewModel.toggleFavorite(tvShow)
54                     lifecycleScope.launch {
55                         updateFavIcon(favoriteViewModel.isFavorite(tvShow))
56                     }
57                 }
58             }
59         }
60
61         return binding.root
62     }
63
64     private fun updateFavIcon(isFav: Boolean) {
65         binding.favIconButton.setImageResource(

```

66	if (isFav) R.drawable.ic_favorite else
67	R.drawable.ic_favorite_border
68	)
69	val descRes = if (isFav) R.string.remove_from_favorite
70	else R.string.add_to_favorite
71	binding.favIconButton.contentDescription =
72	getString(descRes)
73	}
74	}

Tabel 27. Source Code Jawaban Soal No.1 Modul 5

#### 4. DramaViewModel

1	package com.example.modul5
2	
3	import android.util.Log
4	import androidx.lifecycle.ViewModel
5	import androidx.lifecycle.ViewModelScope
6	import kotlinx.coroutines.flow.MutableStateFlow
7	import kotlinx.coroutines.flow.StateFlow
8	import kotlinx.coroutines.launch
9	
10	class DramaViewModel(private val repository: TmdbRepository) :
11	ViewModel() {
12	
13	private val _tvShows =
14	MutableStateFlow<List<TvShow>>(emptyList())
15	val tvShows: StateFlow<List<TvShow>> = _tvShows
16	
17	init {
18	loadTvShows()
19	}
20	
21	private fun loadTvShows() {
22	viewModelScope.launch {
23	try {
24	val response = repository.getChineseTvShows()
25	_tvShows.value = response.results
26	Log.d("DramaViewModel", "Loaded
27	\${response.results.size} Chinese dramas.")
28	} catch (e: Exception) {
29	Log.e("DramaViewModel", "Failed to load data",
30	e)
31	}
32	}
33	}
34	}

Tabel 28. Source Code Jawaban Soal No.1 Modul 5

#### 5. DramaViewModelFactory

1	package	com.example.modul5
2		
3	import	androidx.lifecycle.ViewModel
4	import	androidx.lifecycle.ViewModelProvider
5		
6	class	DramaViewModelFactory(private val repository:
7	TmdbRepository)	: ViewModelProvider.Factory {
8	override fun <T : ViewModel> create(modelClass: Class<T>):	
9	T	{
10	if	
11	(modelClass.isAssignableFrom(DramaViewModel::class.java))	{
12	return	DramaViewModel(repository) as T
13	}	
14	throw	IllegalArgumentException("Unknown ViewModel
15	class")	
16	}	
17	}	

Tabel 29. Source Code Jawaban Soal No.1 Modul 5

## 6. FavoriteDao.kt

1	package	com.example.modul5
2		
3	import	androidx.room.*
4	import	kotlinx.coroutines.flow.Flow
5		
6	@Dao	
7	interface	FavoriteDao {
8		
9	@Insert(onConflict = OnConflictStrategy.REPLACE)	
10	suspend fun addToFavorite(tvShow: FavoriteTvShow)	
11		
12	@Delete	
13	suspend fun removeFromFavorite(tvShow: FavoriteTvShow)	
14		
15	@Query("SELECT * FROM favorite_tv_shows")	
16	fun getAllFavorites(): Flow<List<FavoriteTvShow>>	
17		
18	@Query("SELECT * FROM favorite_tv_shows WHERE id = :id LIMIT	
19	1")	
20	suspend fun getFavoriteById(id: Int): FavoriteTvShow?	
21	}	

Tabel 30. Source Code Jawaban Soal No.1 Modul 5

## 7. FavoriteFragment,kt

1	package	com.example.modul5
2		
3	import	android.os.Bundle
4	import	android.view.LayoutInflater
5	import	android.view.View
6	import	android.view.ViewGroup

```

7 import androidx.fragment.app.Fragment
8 import androidx.fragment.app.viewModels
9 import androidx.lifecycle.ViewModelProvider
10 import androidx.lifecycle.lifecycleScope
11 import androidx.recyclerview.widget.LinearLayoutManager
12 import com.example.modul5.databinding.FragmentFavoriteBinding
13 import kotlinx.coroutines.flow.collectLatest
14 import kotlinx.coroutines.launch
15
16 class FavoriteFragment : Fragment() {
17
18     private lateinit var binding: FragmentFavoriteBinding
19     private val favoriteViewModel: FavoriteViewModel by
20 viewModels
21
22     ViewModelProvider.AndroidViewModelFactory.getInstance(requireA
23 ctivity().application)
24 }
25
26     private lateinit var adapter: ListDramaAdapter
27
28     override fun onCreateView(
29         inflater: LayoutInflater, container: ViewGroup?,
30 savedInstanceState: Bundle?
31     ): View {
32         binding = FragmentFavoriteBinding.inflate(inflater,
33 container,
34         false)
35         return binding.root
36     }
37
38     override fun onViewCreated(view: View, savedInstanceState:
39 Bundle?) {
40         adapter = ListDramaAdapter(emptyList()) { selected ->
41             val fragment = DetailFragment().apply {
42                 arguments = Bundle().apply {
43                     putParcelable("tvshow", selected)
44                 }
45             }
46             parentFragmentManager.beginTransaction()
47                 .replace(R.id.fragment_container, fragment)
48                 .addToBackStack(null)
49                 .commit()
50         }
51
52         binding.recyclerView.apply
53         layoutManager
54         =
55         LinearLayoutManager(requireContext())
56         adapter = this@FavoriteFragment.adapter
57     }
58
59     binding.backButton.setOnClickListener {
60         parentFragmentManager.popBackStack()
61     }
62 }

```

59	}
60	
61	viewLifecycleOwner.lifecycleScope.launch {
62	favoriteViewModel.favorites.collectLatest {
63	favoriteList ->
64	val tvShowList = favoriteList.map {
65	it.toTvShow() }
66	adapter.updateData(tvShowList)
67	}
68	}
69	}
70	}

Tabel 31. Source Code Jawaban Soal No.1 Modul 5

## 8. FavoriteRepository

1	package	com.example.modul5
2		
3	import	android.util.Log
4	import	kotlinx.coroutines.flow.Flow
5		
6	class FavoriteRepository(private val dao: FavoriteDao) {	
7		
8	fun getFavorites(): Flow<List<FavoriteTvShow>> =	
9	dao.getAllFavorites()	
10		
11	suspend fun isFavorite(id: Int): Boolean =	
12	dao.getFavoriteById(id) != null	
13		
14	suspend fun add(tvShow: FavoriteTvShow) {	
15	dao.addToFavorite(tvShow)	
16	}	
17		
18	suspend fun remove(tvShow: FavoriteTvShow) {	
19	dao.removeFromFavorite(tvShow)	
20	}	
21	}	

Tabel 32. Source Code Jawaban Soal No.1 Modul 5

## 9. FavoriteTvShow.kt

1	package	com.example.modul5
2		
3	import	androidx.room.Entity
4	import	androidx.room.PrimaryKey
5		
6	@Entity(tableName =	"favorite_tv_shows")
7	data class FavoriteTvShow(	
8	@PrimaryKey val id:	Int,
9	val name:	String,
10	val posterPath:	String?,
11	val firstAirDate:	String,

```

12     val rating: Double,
13     val overview: String
14 )
15 fun FavoriteTvShow.toTvShow(): TvShow {
16     return TvShow(
17         id = this.id,
18         name = this.name,
19         posterPath = this.posterPath,
20         rating = this.rating,
21         firstAirDate = this.firstAirDate,
22         overview = this.overview
23     )
24 }

```

Tabel 33. Source Code Jawaban Soal No.1 Modul 5

## 10. FavoriteViewModel.kt

```

1 package com.example.modul5
2
3 import android.app.Application
4 import android.util.Log
5 import androidx.lifecycle.AndroidViewModel
6 import androidx.lifecycle.ViewModelScope
7 import kotlinx.coroutines.flow.*
8 import kotlinx.coroutines.launch
9
10 class FavoriteViewModel(application: Application) :
11     AndroidViewModel(application) {
12
13     private val db = AppDatabase.getDatabase(application)
14     private val repository =
15         FavoriteRepository(db.favoriteDao())
16
17     val favorites: StateFlow<List<FavoriteTvShow>> =
18         repository.getFavorites()
19             .stateIn(viewModelScope,
20                 SharingStarted.WhileSubscribed(5000), emptyList())
21
22     fun toggleFavorite(tvShow: TvShow) {
23         viewModelScope.launch {
24             val fav = FavoriteTvShow(
25                 id = tvShow.id,
26                 name = tvShow.name,
27                 posterPath = tvShow.posterPath,
28                 firstAirDate = tvShow.firstAirDate,
29                 rating = tvShow.rating,
30                 overview = tvShow.overview
31             )
32             if (repository.isFavorite(tvShow.id)) {
33                 repository.remove(fav)
34             } else {
35                 repository.add(fav)
36             }
37         }
38     }
39 }

```

36	}
37	}
38	}
39	
40	suspend fun isFavorite(tvShow: TvShow): Boolean {
41	return repository.isFavorite(tvShow.id)
42	}
43	}

Tabel 34. Source Code Jawaban Soal No.1 Modul 5

## 11. HomeFragment.kt

1	package	com.example.modul5
2		
3	import	android.os.Bundle
4	import	android.view.LayoutInflater
5	import	android.view.View
6	import	android.view.ViewGroup
7	import	androidx.fragment.app.Fragment
8	import	androidx.fragment.app.viewModels
9	import	androidx.lifecycle.lifecycleScope
10	import	androidx.recyclerview.widget.LinearLayoutManager
11	import	com.example.modul5.databinding.FragmentHomeBinding
12	import	kotlinx.coroutines.flow.collectLatest
13	import	kotlinx.coroutines.launch
14		
15	class	HomeFragment : Fragment() {
16		
17	private lateinit var	binding: FragmentHomeBinding
18	private val viewModel:	DramaViewModel by viewModels {
19		
20	DramaViewModelFactory(TmdbRepository(NetworkModule.apiService))	
21	}	
22	private lateinit var	adapter: ListDramaAdapter
23		
24	override fun onCreateView(inflater: LayoutInflater,	
25	container: ViewGroup?, savedInstanceState: Bundle?): View {	
26	binding =	FragmentHomeBinding.inflate(inflater,
27	container,	false)
28	return	binding.root
29	}	
30		
31	override fun onViewCreated(view: View, savedInstanceState:	
32	Bundle?)	{
33	adapter = ListDramaAdapter(emptyList()) { selectedTvShow	
34	->	
35	val fragment =	DetailFragment().apply {
36	arguments =	Bundle().apply {
37	putParcelable("tvshow",	selectedTvShow)
38	}	
39	}	
40	parentFragmentManager.beginTransaction()	



```

41         .replace(R.id.fragment_container, fragment)
42         .addToBackStack(null)
43         .commit()
44     }
45
46     binding.recyclerView.layoutManager =
47     LinearLayoutManager(requireContext())
48     binding.recyclerView.adapter = adapter
49
50     viewLifecycleOwner.lifecycleScope.launch {
51         viewModel.tvShows.collectLatest { tvList ->
52             adapter.updateData(tvList)
53         }
54     }
55
56     binding.buttonFavorite.setOnClickListener {
57         parentFragmentManager.beginTransaction()
58             .replace(R.id.fragment_container,
59             FavoriteFragment())
60             .addToBackStack(null)
61             .commit()
62     }
63 }
64 }

```

*Tabel 35. Source Code Jawaban Soal No.1 Modul 5*

## 12. ListDramaAdapter.kt

```

1 package com.example.modul5
2
3 import android.content.Intent
4 import android.net.Uri
5 import android.view.LayoutInflater
6 import android.view.ViewGroup
7 import androidx.recyclerview.widget.RecyclerView
8 import com.example.modul5.databinding.ItemDramaBinding
9
10 class ListDramaAdapter(
11     private var list: List<TvShow>,
12     private val onClick: (TvShow) -> Unit
13 ) : RecyclerView.Adapter<ListDramaAdapter.MyViewHolder>() {
14
15     inner class MyViewHolder(val binding: ItemDramaBinding) :
16     RecyclerView.ViewHolder(binding.root)
17
18     override fun onCreateViewHolder(parent: ViewGroup, viewType:
19     Int): MyViewHolder {
20         val binding =
21         ItemDramaBinding.inflate(LayoutInflater.from(parent.context),
22         parent, false)
23         return MyViewHolder(binding)
24     }

```

```

25
26     override fun getItemCount(): Int = list.size
27
28     override fun onBindViewHolder(holder: MyViewHolder,
29 position: Int) {
30         val tvShow = list[position]
31         val context = holder.binding.root.context
32
33         holder.binding.tvShow = tvShow
34
35         holder.binding.buttonDetail.setOnClickListener {
36             onDetailClick(tvShow)
37         }
38
39         holder.binding.buttonLink.setOnClickListener {
40             val intent = Intent(Intent.ACTION_VIEW,
41 Uri.parse("https://www.themoviedb.org/tv/${tvShow.id}"))
42             context.startActivity(intent)
43         }
44     }
45
46     fun updateData(newList: List<TvShow>) {
47         list = newList
48         notifyDataSetChanged()
49     }
50 }

```

Tabel 36. Source Code Jawaban Soal No.1 Modul 5

### 13. MainActivity.kt

```

1 package com.example.modul5
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import com.example.modul5.databinding.ActivityMainBinding
6
7 class MainActivity : AppCompatActivity() {
8
9     private lateinit var binding: ActivityMainBinding
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         binding = ActivityMainBinding.inflate(layoutInflater)
14         setContentView(binding.root)
15
16         if (savedInstanceState == null) {
17             supportFragmentManager.beginTransaction()
18                 .replace(R.id.fragment_container,
19 HomeFragment())
20                 .commit()
21         }
22

```

23	}
	}

Tabel 37. Source Code Jawaban Soal No.1 Modul 5

#### 14. NetworkModule.kt

1	package	com.example.modul5
2		
3	import	
4	com.jakewharton.retrofit2.converter.kotlinx.serialization.asCo	
5	nverterFactory	
6	import	kotlinx.serialization.json.Json
7	import	okhttp3.MediaType.Companion.toMediaType
8	import	retrofit2.Retrofit
9		
10	object	NetworkModule {
11		
12	private const val	BASE_URL = "https://api.themoviedb.org/3/"
13		
14	val	apiService: TmdbApiService by lazy {
15	val	contentType = "application/json".toMediaType()
16		
17	val	json = Json {
18	ignoreUnknownKeys	= true
19	}	
20		
21	Retrofit.Builder()	
22	.baseUrl	(BASE_URL)
23		
24	.addConverterFactory	(json.asConverterFactory(contentType))
25	.build()	
26	.create	(TmdbApiService::class.java)
27	}	
28	}	

Tabel 38. Source Code Jawaban Soal No.1 Modul 5

#### 15. TmdbApiService.kt

1	package	com.example.modul5
2		
3	import	retrofit2.http.GET
4	import	retrofit2.http.Query
5		
6	interface	TmdbApiService {
7	@GET("discover/tv")	
8	suspend fun	getChineseTvShows (
9	@Query("api_key")	apiKey: String,
10	@Query("with_original_language")	language: String =
11	"zh",	
12	@Query("sort_by")	sortBy: String = "popularity.desc",
13	@Query("first_air_date.gte")	startDate: String = "2020-
14	01-01",	

15	@Query("first_air_date.lte") endDate: String = "2025-05-
16	31"
17	): TvShowResponse
18	}

Tabel 39. Source Code Jawaban Soal No.1 Modul 5

## 16. TmdbRepository.kt

1	package com.example.modul5
2	
3	class TmdbRepository(private val apiService: TmdbApiService) {
4	suspend fun getChineseTvShows(): TvShowResponse {
5	return apiService.getChineseTvShows(
6	apiKey = "36854acbbff7b761e0f196e36d1fc1a9",
7	startDate = "2023-01-01",
8	endDate = "2025-05-31"
9	)
10	}
11	}

Tabel 40. Source Code Jawaban Soal No.1 Modul 5

## 17. TvShow.kt

1	package com.example.modul5
2	
3	import android.os.Parcelable
4	import kotlinx.parcelize.Parcelize
5	import kotlinx.serialization.SerialName
6	import kotlinx.serialization.Serializable
7	
8	@Parcelize
9	@Serializable
10	data class TvShow(
11	val id: Int,
12	val name: String,
13	@SerialName("poster_path") val posterPath: String?,
14	@SerialName("vote_average") val rating: Double,
15	@SerialName("first_air_date") val firstAirDate: String,
16	val overview: String
17	) : Parcelable

Tabel 41. Source Code Jawaban Soal No.1 Modul 5

## 18. TvShowResponse.kt

1	package com.example.modul5
2	
3	import kotlinx.serialization.Serializable
4	
5	@Serializable
6	data class TvShowResponse(
7	val results: List<TvShow>
8	)

Tabel 42. Source Code Jawaban Soal No.1 Modul 5

## 19. activity\_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3 xmlns:android="http://schemas.android.com/apk/res/android"
4 xmlns:app="http://schemas.android.com/apk/res-auto"
5 xmlns:tools="http://schemas.android.com/tools"
6 android:layout_width="match_parent"
7 android:layout_height="match_parent">
8
9 <FrameLayout
10 android:id="@+id/fragment_container"
11 android:layout_width="match_parent"
12 android:layout_height="match_parent" />
13 </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 43. Source Code Jawaban Soal No.1 Modul 5

## 20. fragment\_detail.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <layout
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
7 <data>
8 <variable
9 name="tvShow"
10 type="com.example.modul5.TvShow" />
11 </data>
12
13 <ScrollView
14 android:layout_width="match_parent"
15 android:layout_height="match_parent"
16 tools:context=".DetailFragment">
17
18 <LinearLayout
19 android:orientation="vertical"
20 android:padding="16dp"
21 android:layout_width="match_parent"
22 android:layout_height="wrap_content"
23 android:gravity="center_horizontal"
24 android:background="@android:color/white">
25
26 <ImageView
27 android:id="@+id/detail_image"
28 android:layout_width="match_parent"
29 android:layout_height="500dp"
30 android:scaleType="centerCrop"
31
32 android:contentDescription="@string/detail_image"
33 app:imageUrl="@{tvShow.posterPath} != null ?
```

```

34 "https://image.tmdb.org/t/p/w500" + tvShow.posterPath : null}'
35 />
36
37 <TextView
38     android:id="@+id/title_text"
39     android:layout_width="wrap_content"
40     android:layout_height="wrap_content"
41     android:text="@{tvShow.name}"
42     android:textSize="22sp"
43     android:textStyle="bold"
44     android:paddingTop="12dp"
45     android:layout_gravity="center_horizontal" />
46
47 <TextView
48     android:id="@+id/year_genre_text"
49     android:layout_width="wrap_content"
50     android:layout_height="wrap_content"
51     android:text="@{tvShow.firstAirDate}"
52     android:textSize="16sp"
53     android:paddingTop="4dp"
54     android:layout_gravity="center_horizontal" />
55
56 <TextView
57     android:id="@+id/rating_text"
58     android:layout_width="wrap_content"
59     android:layout_height="wrap_content"
60     android:text='@{"Rating:           " +
61 String.valueOf(tvShow.rating)}'
62     android:textSize="16sp"
63     android:paddingTop="4dp"
64     android:layout_gravity="center_horizontal" />
65
66 <ImageButton
67     android:id="@+id/favIconButton"
68     android:layout_width="60dp"
69     android:layout_height="60dp"
70     android:layout_marginTop="12dp"
71     android:layout_gravity="center_horizontal"
72
73     android:background="?attr/selectableItemBackgroundBorderless"
74
75     android:contentDescription="@string/add_to_favorite"
76     android:src="@drawable/ic_favorite_border" />
77
78 <LinearLayout
79     android:layout_width="wrap_content"
80     android:layout_height="wrap_content"
81     android:orientation="horizontal"
82     android:gravity="center"
83     android:layout_marginTop="16dp">
84
85     <Button

```

86	android:id="@+id/openLinkButton"
87	android:layout_width="wrap_content"
88	android:layout_height="wrap_content"
89	android:text="@string/watch_here"
90	android:textColor="@android:color/white"
91	android:backgroundTint="@color/primary"
92	android:layout_marginEnd="8dp" />
93	
94	<Button
95	android:id="@+id/backButton"
96	android:layout_width="wrap_content"
97	android:layout_height="wrap_content"
98	android:text="@string/button_back"
99	android:textColor="@android:color/white"
100	android:backgroundTint="@color/primary" />
101	</LinearLayout>
102	</LinearLayout>
103	</ScrollView>
104	</layout>

Tabel 44. Source Code Jawaban Soal No.1 Modul 5

## 21. fragment\_favorite.xml

1	<layout
2	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto">
4	
5	<data>
6	</data>
7	
8	<androidx.constraintlayout.widget.ConstraintLayout
9	android:layout_width="match_parent"
10	android:layout_height="match_parent">
11	
12	<LinearLayout
13	android:id="@+id/headerBar"
14	android:layout_width="0dp"
15	android:layout_height="wrap_content"
16	android:orientation="horizontal"
17	android:padding="16dp"
18	android:background="#2196F3"
19	app:layout_constraintTop_toTopOf="parent"
20	app:layout_constraintStart_toStartOf="parent"
21	app:layout_constraintEnd_toEndOf="parent">
22	
23	<Button
24	android:id="@+id/backButton"
25	android:layout_width="wrap_content"
26	android:layout_height="wrap_content"
27	android:text="@string/button_back"
28	android:textColor="@android:color/white"
29	android:background="@android:color/transparent"

```

30 />
31
32         <TextView
33             android:id="@+id/titleText"
34             android:layout_width="180dp"
35             android:layout_height="wrap_content"
36             android:layout_gravity="center_vertical"
37             android:layout_marginStart="16dp"
38             android:text="@string/favorite_title"
39             android:textColor="@android:color/white"
40             android:textSize="18sp"
41             android:textStyle="bold"
42         </LinearLayout>
43
44         <androidx.recyclerview.widget.RecyclerView
45             android:id="@+id/recyclerView"
46             android:layout_width="0dp"
47             android:layout_height="0dp"
48             app:layout_constraintTop_toBottomOf="@id/headerBar"
49             app:layout_constraintBottom_toBottomOf="parent"
50             app:layout_constraintStart_toStartOf="parent"
51             app:layout_constraintEnd_toEndOf="parent"/>
52
53     </androidx.constraintlayout.widget.ConstraintLayout>
54 </layout>

```

*Tabel 45. Source Code Jawaban Soal No.1 Modul 5*

## 22. fragment\_home.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent">
7
8      <LinearLayout
9          android:id="@+id/headerBar"
10         android:layout_width="0dp"
11         android:layout_height="wrap_content"
12         android:orientation="horizontal"
13         android:background="#2196F3"
14         android:padding="16dp"
15         app:layout_constraintTop_toTopOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintEnd_toEndOf="parent">
18
19         <TextView
20             android:id="@+id/appTitleText"
21             android:layout_width="0dp"
22             android:layout_height="wrap_content"
23             android:layout_weight="1"

```



24	android:text="@string/app_name"	
25	android:textSize="18sp"	
26	android:textStyle="bold"	
27	android:textColor="@android:color/white"	/>
28		
29	<Button	
30	android:id="@+id/buttonFavorite"	
31	android:layout_width="wrap_content"	
32	android:layout_height="wrap_content"	
33	android:text="@string/go_to_favorites"	
34	android:textColor="@android:color/white"	
35	android:background="@android:color/transparent"	/>
36	</LinearLayout>	
37		
38	<androidx.recyclerview.widget.RecyclerView	
39	android:id="@+id/recyclerView"	
40	android:layout_width="0dp"	
41	android:layout_height="0dp"	
42	app:layout_constraintTop_toBottomOf="@id/headerBar"	
43	app:layout_constraintBottom_toBottomOf="parent"	
44	app:layout_constraintStart_toStartOf="parent"	
45	app:layout_constraintEnd_toEndOf="parent"	/>
46	</androidx.constraintlayout.widget.ConstraintLayout>	

*Tabel 46. Source Code Jawaban Soal No.1 Modul 5*

## 23. item\_drama.xml

1	<?xml	version="1.0"	encoding="utf-8"?">
2	<layout		
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			
7	<data>		
8	<variable		
9	name="tvShow"		
10	type="com.example.modul5.TvShow"		/>
11	</data>		
12			
13	<androidx.cardview.widget.CardView		
14	android:layout_width="match_parent"		
15	android:layout_height="wrap_content"		
16	android:layout_margin="8dp"		
17	app:cardCornerRadius="12dp"		
18	app:cardElevation="4dp">		
19			
20	<androidx.constraintlayout.widget.ConstraintLayout		
21	android:layout_width="match_parent"		
22	android:layout_height="wrap_content"		
23	android:padding="12dp">		
24			
25	<ImageView		

```

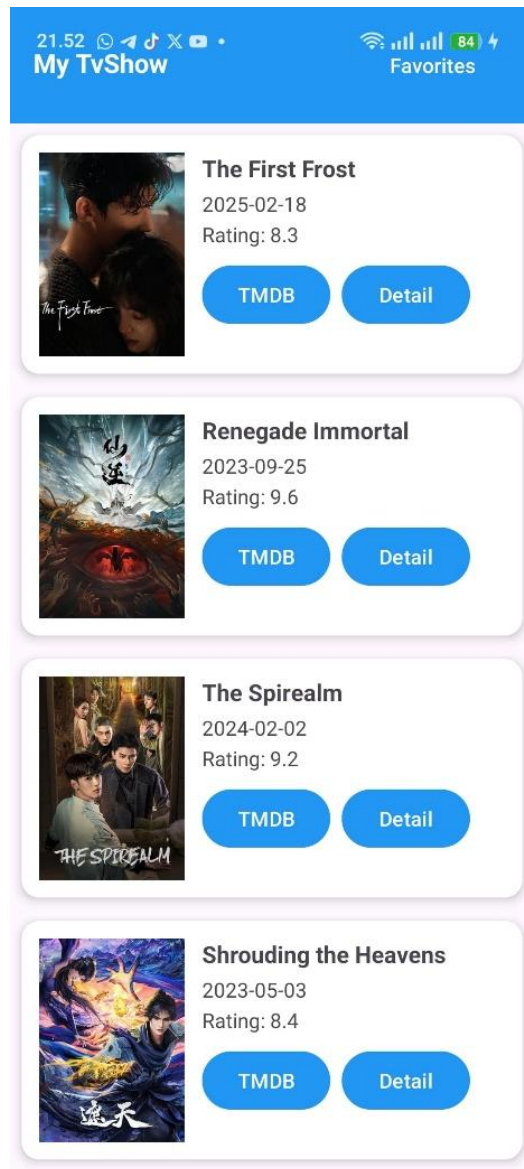
26         android:id="@+id/imageView"
27         android:layout_width="100dp"
28         android:layout_height="140dp"
29         android:scaleType="centerCrop"
30
31     android:contentDescription="@string/detail_image"
32
33     app:imageUrl='{ "https://image.tmdb.org/t/p/w500"           +
34     tvShow.posterPath } '
35         app:layout_constraintStart_toStartOf="parent"
36         app:layout_constraintTop_toTopOf="parent"
37
38     app:layout_constraintBottom_toBottomOf="parent"           />
39
40     <TextView
41         android:id="@+id/titleText"
42         android:layout_width="0dp"
43         android:layout_height="wrap_content"
44         android:text="@{tvShow.name}"
45         android:textStyle="bold"
46         android:textSize="16sp"
47
48     app:layout_constraintTop_toTopOf="@id/imageView"
49
50     app:layout_constraintStart_toEndOf="@id/imageView"
51         app:layout_constraintEnd_toEndOf="parent"
52         android:layout_marginStart="12dp"           />
53
54     <TextView
55         android:id="@+id/yearText"
56         android:layout_width="0dp"
57         android:layout_height="wrap_content"
58         android:text="@{tvShow.firstAirDate}"
59         android:textSize="14sp"
60
61     app:layout_constraintTop_toBottomOf="@id/titleText"
62
63     app:layout_constraintStart_toStartOf="@+id/titleText"
64         app:layout_constraintEnd_toEndOf="parent"
65         android:layout_marginTop="4dp"           />
66
67     <TextView
68         android:id="@+id/ratingText"
69         android:layout_width="0dp"
70         android:layout_height="wrap_content"
71         android:text='{ "Rating:           "           +
72     String.valueOf(tvShow.rating) } '
73
74     app:layout_constraintTop_toBottomOf="@id/yearText"
75
76     app:layout_constraintStart_toStartOf="@+id/titleText"
77         app:layout_constraintEnd_toEndOf="parent"

```

78	android:layout_marginTop="2dp"	/>
79		
80	<Button	
81	android:id="@+id/buttonLink"	
82	android:layout_width="wrap_content"	
83	android:layout_height="wrap_content"	
84	android:text="@string/watch_here"	
85	android:textColor="@android:color/white"	
86	android:backgroundTint="@color/primary"	
87		
88	app:layout_constraintTop_toBottomOf="@id/ratingText"	
89		
90	app:layout_constraintStart_toStartOf="@+id/titleText"	
91	android:layout_marginTop="8dp"	/>
92		
93	<Button	
94	android:id="@+id/buttonDetail"	
95	android:layout_width="wrap_content"	
96	android:layout_height="wrap_content"	
97	android:text="@string/title_detail"	
98	android:textColor="@android:color/white"	
99	android:backgroundTint="@color/primary"	
100		
101	app:layout_constraintTop_toTopOf="@id/buttonLink"	
102		
103	app:layout_constraintStart_toEndOf="@id/buttonLink"	
104	android:layout_marginStart="8dp"	/>
105		
106	</androidx.constraintlayout.widget.ConstraintLayout>	
107	</androidx.cardview.widget.CardView>	
108	</layout>	

Tabel 47. Source Code Jawaban Soal No.1 Modul 5

## B. Output Program



Gambar 13. Screenshot Hasil Jawaban Soal No. 1 Modul 5



## The Spirealm

2024-02-02

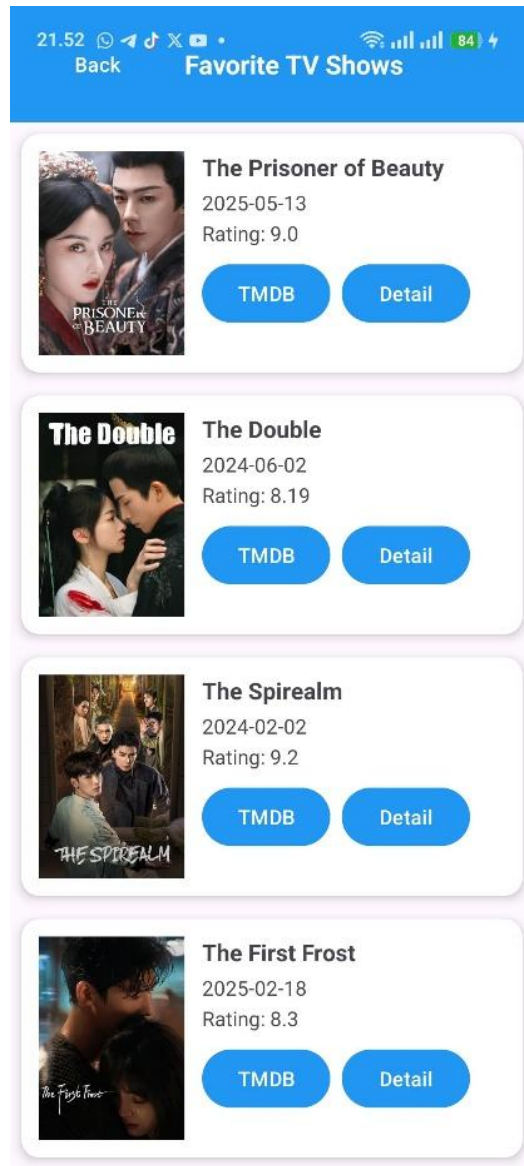
Rating: 9.2



TMDB

Back

Gambar 14. Screenshot Hasil Jawaban Soal No. 1 Modul 5



Gambar 15. Screenshot Hasil Jawaban Soal No. 1 Modul 5

### C. Pembahasan

- **AppDatabase.kt:**

AppDatabase ini intinya adalah cetak biru untuk database lokal sebuah aplikasi. Di sinilah ditentukan data apa saja yang akan disimpan, seperti daftar acara TV favorit, dan juga versi database-nya. Ada juga bagian yang bertugas sebagai "penjaga" database, yang disebut DAO (Data Access Object); dia ini yang punya semua perintah untuk menyimpan, menghapus, atau mencari data. Nah, yang penting lagi, AppDatabase ini memastikan cuma ada satu

database yang aktif di seluruh aplikasi. Ini penting banget agar aplikasi efisien dan tidak bantrok saat banyak bagian program mau pakai database bersamaan. Jadi, jika akses ke database dibutuhkan, tinggal panggil saja fungsi khusus yang sudah disediakan, dan dia akan otomatis memberikan akses atau bahkan membuat database baru jika belum ada.

- **BindingAdapter.kt**

Secara sederhana, kode ini membuat sebuah aturan khusus di mana setiap kali ada elemen ImageView di tata letak XML yang memiliki atribut imageUrl (yang dibuat sendiri), maka fungsi loadImage ini akan dipanggil. Di dalam fungsi loadImage, ia menerima dua hal: view (yaitu ImageView itu sendiri) dan url (alamat gambar dari internet). Kode ini memeriksa apakah url gambar tidak kosong. Jika ada url, ia akan menggunakan pustaka Glide untuk mengunduh gambar dari url tersebut dan langsung menampilkannya ke ImageView. Ini artinya, pengembang tidak perlu lagi menulis kode Java atau Kotlin yang rumit untuk memuat gambar dari internet setiap kali mereka ingin menampilkan gambar; cukup dengan menambahkan atribut imageUrl di XML.

- **DetailFragment.kt**

DetailFragment adalah komponen antarmuka pengguna (UI) yang dirancang untuk menampilkan informasi lengkap dari sebuah acara TV. Saat fragment ini dimuat, ia akan mengambil data acara TV yang diterima, lalu menampilkan semua detailnya, seperti judul, deskripsi, dan gambar, ke tampilan. Fragment ini memiliki beberapa fitur utama: ada tombol untuk membuka tautan yang akan mengarahkan pengguna ke halaman acara TV di situs The Movie Database melalui browser, tombol kembali untuk navigasi ke layar sebelumnya, serta tombol favorit. Tombol favorit ini sangat interaktif, karena DetailFragment berkomunikasi dengan FavoriteViewModel untuk mengecek status favorit acara TV tersebut. Berdasarkan status ini, ikon hati akan berubah menjadi penuh jika sudah difavoritkan atau berongga jika belum. Jika pengguna mengklik ikon hati, status favorit acara TV akan diubah (ditambah atau dihapus dari daftar favorit), dan ikon akan langsung diperbarui secara otomatis. Jadi, DetailFragment ini bertanggung jawab penuh dalam menampilkan detail acara TV, membuka tautan terkait, memfasilitasi navigasi, dan mengelola status favorit acara TV tersebut.

- **DramaViewmodel.kt**

DramaViewModel adalah sebuah ViewModel yang bertanggung jawab untuk mengambil dan mengelola daftar acara TV drama Tiongkok. ViewModel ini dirancang agar data tetap ada meskipun terjadi perubahan konfigurasi pada perangkat. Saat DramaViewModel pertama kali dibuat, ia akan langsung memanggil fungsi loadTvShows(). Fungsi ini bekerja di balik layar, menggunakan viewModelScope untuk menjalankan operasi jaringan. Ia akan memanggil repository (yang fungsinya berhubungan dengan data dari The Movie Database) untuk mendapatkan daftar drama Tiongkok. Jika berhasil, daftar acara TV tersebut akan disimpan dan siap ditampilkan ke antarmuka pengguna. Namun, jika terjadi kesalahan saat mengambil data, pesan kesalahan akan dicatat. Jadi, DramaViewModel ini adalah jembatan antara tampilan aplikasi dan data drama Tiongkok yang diambil dari internet, memastikan data tersedia dan diperbarui dengan lancar.

- **DramaViewmodelFactory.kt**

DramaViewModelFactory ini adalah sebuah pabrik khusus yang tugasnya adalah membuat dan menyediakan DramaViewModel dengan benar. Karena DramaViewModel membutuhkan TmdbRepository (yang bertugas mengambil data dari internet), pabrik ini memastikan bahwa setiap kali DramaViewModel diminta, ia akan dibuat dan langsung dilengkapi dengan TmdbRepository yang dibutuhkan. Jadi, DramaViewModelFactory ini berfungsi seperti manajer produksi yang memastikan DramaViewModel selalu siap pakai dengan semua komponen yang diperlukan.

- **FavoriteDao.kt**

FavoriteDao adalah sebuah antarmuka (interface) yang berfungsi sebagai Data Access Object (DAO) untuk database Room aplikasi. Ini adalah "penjaga" yang berisi semua perintah untuk mengelola data acara TV favorit. Di sini, didefinisikan beberapa operasi dasar: ada fungsi addToFavorite yang bisa menambahkan acara TV ke daftar favorit (atau menggantinya jika sudah ada), removeFromFavorite untuk menghapus acara TV dari daftar favorit, getAllFavorites yang akan memberikan semua daftar acara TV favorit dan akan terus diperbarui secara otomatis jika ada perubahan, serta getFavoriteById untuk mencari apakah



ada acara TV tertentu berdasarkan ID-nya di daftar favorit. Semua fungsi ini dirancang untuk berinteraksi dengan database secara aman dan efisien.

- **FavoriteFragment.kt**

FavoriteFragment adalah bagian antarmuka pengguna yang dirancang untuk menampilkan daftar semua acara TV yang sudah ditandai sebagai favorit. Fragment ini menggunakan RecyclerView untuk menampilkan daftar tersebut secara efisien. Ketika fragment ini dibuat, ia akan menyiapkan tampilan dan adapter untuk RecyclerView agar bisa menampilkan data. Fragment ini juga berinteraksi dengan FavoriteViewModel untuk terus memantau daftar acara TV favorit. Setiap kali ada perubahan pada daftar favorit, tampilan akan otomatis diperbarui. Jika pengguna mengklik salah satu acara TV di daftar favorit, aplikasi akan membuka DetailFragment untuk menampilkan informasi lebih lanjut tentang acara TV yang dipilih tersebut. Ada juga tombol kembali yang memungkinkan pengguna untuk dengan mudah menavigasi ke layar sebelumnya.

- **FavoriteRepository.kt**

FavoriteRepository ini adalah penghubung antara FavoriteViewModel dan database lokal yang sebenarnya, yang diwakili oleh FavoriteDao. Tugas utamanya adalah mengelola semua operasi terkait acara TV favorit. Repository ini menyediakan beberapa fungsi penting: ada getFavorites() yang akan memberikan daftar semua acara TV yang sudah difavoritkan secara terus-menerus (jika ada perubahan, daftar ini otomatis terbaru), isFavorite() untuk mengecek apakah sebuah acara TV sudah ada di daftar favorit berdasarkan ID-nya, add() untuk menambahkan acara TV ke daftar favorit, dan remove() untuk menghapus acara TV dari daftar tersebut. Dengan adanya repository ini, ViewModel tidak perlu tahu detail bagaimana data disimpan atau diambil dari database; ia cukup memanggil fungsi-fungsi yang disediakan oleh FavoriteRepository ini.

- **FavoriteTvShow.kt**

FavoriteTvShow adalah sebuah kelas data yang mewakili satu baris data di dalam database Room aplikasi. Kelas ini ditandai sebagai @Entity dengan nama tabel "favorite\_tv\_shows", yang berarti setiap objek dari kelas ini akan disimpan sebagai entri terpisah di tabel tersebut.

Setiap acara TV favorit memiliki id unik sebagai kunci utama, name, posterPath (jalur gambar poster), firstAirDate (tanggal tayang perdana), rating, dan overview (ringkasan). Selain itu, ada juga fungsi ekstensi toTvShow() yang sangat praktis. Fungsi ini memungkinkan untuk dengan mudah mengubah objek FavoriteTvShow menjadi objek TvShow biasa, yang kemungkinan digunakan di bagian lain aplikasi yang menampilkan data acara TV secara umum.

- **FavoriteViewModel.kt**

FavoriteViewModel adalah sebuah ViewModel yang bertugas mengelola data acara TV favorit di aplikasi. ViewModel ini berkomunikasi dengan AppDatabase untuk mendapatkan akses ke FavoriteRepository, yang selanjutnya akan berinteraksi langsung dengan database. Tugas utamanya adalah menyediakan daftar acara TV favorit secara real-time kepada tampilan, sehingga setiap kali ada perubahan di database, tampilan akan otomatis diperbarui. Selain itu, ViewModel ini juga memiliki fungsi untuk mengubah status favorit sebuah acara TV (menambahkannya ke favorit atau menghapusnya) dan memeriksa apakah suatu acara TV sudah difavoritkan atau belum, semuanya dilakukan dengan efisien di latar belakang tanpa menghambat antarmuka pengguna.

- **HomeFragment.kt**

HomeFragment adalah bagian utama antarmuka pengguna aplikasi yang menampilkan daftar acara TV drama. Fragment ini bertugas untuk menyiapkan tampilan dan RecyclerView yang akan menampilkan daftar drama tersebut. Ia menggunakan DramaViewModel untuk mendapatkan data acara TV, dan setiap kali data baru tersedia atau diperbarui, tampilan daftar akan otomatis ikut berubah. Jika pengguna mengetuk salah satu acara TV dalam daftar, aplikasi akan langsung berpindah ke DetailFragment untuk menunjukkan informasi lebih lengkap tentang acara TV yang dipilih. Selain itu, ada juga tombol khusus yang memungkinkan pengguna untuk dengan mudah berpindah ke layar FavoriteFragment, tempat daftar acara TV favorit disimpan.

- **ListDramaAdapter.kt**

ListDramaAdapter ini adalah komponen penting yang bertugas menampilkan daftar acara TV dalam bentuk daftar yang bisa digulir di aplikasi. Ia mengambil daftar acara TV dan mengatur bagaimana setiap item dalam daftar tersebut akan terlihat. Untuk setiap acara TV, adapter ini akan membuat tampilannya, mengisi data seperti judul dan poster, lalu menyiapkan dua tombol interaktif. Tombol pertama, "Detail", saat diklik akan memicu aksi untuk menampilkan detail lengkap acara TV tersebut di layar lain. Tombol kedua, "Link", akan membuka halaman acara TV terkait di situs The Movie Database melalui browser web perangkat. Adapter ini juga dilengkapi dengan fungsi `updateData` yang memungkinkan daftar acara TV diperbarui secara efisien kapan pun ada data baru, sehingga tampilan akan selalu segar dan sesuai.

- **MainActivity.kt**

MainActivity adalah aktivitas utama atau bisa dibilang titik masuk pertama kali aplikasi dijalankan. Saat aplikasi dibuka, MainActivity ini akan bertanggung jawab untuk mengatur tata letak utama aplikasi dan secara otomatis menampilkan HomeFragment sebagai tampilan awal. Jika aplikasi ini dibuka lagi setelah sebelumnya ditutup, MainActivity akan memastikan bahwa HomeFragment tidak dibuat ulang secara berlebihan, sehingga pengalaman pengguna tetap lancar.

- **NetworkModule.kt**

NetworkModule adalah sebuah objek yang bertugas untuk menyiapkan segala sesuatu yang dibutuhkan aplikasi agar bisa berkomunikasi dengan API The Movie Database (TMDB) di internet. Modul ini menentukan alamat dasar (`BASE_URL`) dari API tersebut. Bagian utamanya adalah `apiService` yang dibuat secara lazy, artinya ia hanya akan dibuat saat pertama kali dibutuhkan. Di dalamnya, digunakan pustaka Retrofit untuk membangun koneksi ke API TMDB dan Kotlinx Serialization untuk mengubah data dari JSON yang diterima API menjadi objek data yang bisa dimengerti aplikasi. Pengaturan `ignoreUnknownKeys = true` juga ditambahkan untuk memastikan aplikasi tidak akan error jika ada data yang tidak dikenal dari API. Jadi, NetworkModule ini adalah jembatan utama yang memungkinkan aplikasi mengambil data acara TV dari internet.

- **TmdbApiService.kt**

TmdbApiService ini adalah sebuah antarmuka yang menjelaskan bagaimana aplikasi akan berkomunikasi dengan API The Movie Database (TMDB) untuk mengambil data acara TV. Di sini, didefinisikan satu fungsi utama bernama `getChineseTvShows`. Fungsi ini adalah permintaan untuk mendapatkan daftar acara TV Tiongkok (`with_original_language=zh`) yang diurutkan berdasarkan popularitas (`popularity.desc`), serta sudah tayang antara awal tahun 2020 hingga akhir Mei 2025. Yang terpenting, setiap permintaan harus menyertakan `api_key` untuk otentikasi. Jadi, TmdbApiService ini seperti sebuah "menu" yang mendefinisikan permintaan apa saja yang bisa aplikasi kirimkan ke server TMDB.

- **TmdbRepository.kt**

TmdbRepository ini adalah sebuah kelas yang berfungsi sebagai jembatan antara ViewModel aplikasi dan TmdbApiService yang berkomunikasi dengan API The Movie Database. Tugas utamanya adalah menyediakan data acara TV yang bersih dan mudah digunakan untuk ViewModel, tanpa perlu ViewModel tahu bagaimana cara sebenarnya data itu diambil dari internet. Di sini, ada fungsi `getChineseTvShows()` yang saat dipanggil akan langsung meminta data drama Tiongkok dari API. Fungsi ini sudah dilengkapi dengan `apiKey` yang dibutuhkan untuk akses ke API, serta menentukan rentang tanggal tayang perdana yaitu dari 1 Januari 2023 hingga 31 Mei 2025. Jadi, TmdbRepository ini memastikan bahwa data dari internet bisa didapatkan dengan mudah dan terorganisir untuk bagian aplikasi lainnya.

- **TvShow.kt**

TvShow adalah sebuah kelas data yang merepresentasikan informasi detail dari satu acara TV. Kelas ini dibuat agar mudah dikirim antar komponen Android, berkat anotasi `@Parcelize`. Selain itu, dengan anotasi `@Serializable`, data TvShow ini bisa dengan mudah diubah dari atau ke format seperti JSON, yang umum digunakan saat berkomunikasi dengan API. Setiap objek TvShow memiliki beberapa properti penting seperti `id` (pengenal unik), `name` (judul), `posterPath` (lokasi gambar poster), `rating` (rata-rata nilai ulasan), `firstAirDate` (tanggal tayang perdana), dan `overview` (deskripsi singkat). Anotasi `@SerializedName` digunakan untuk mencocokkan nama properti dengan nama yang diterima dari API, seperti `poster_path` yang menjadi `posterPath` di kode.

- **TvShowResponse.kt**

TvShowResponse adalah sebuah kelas data yang berfungsi sebagai wadah untuk menampung hasil dari permintaan API yang berisi daftar acara TV. Kelas ini ditandai dengan `@Serializable`, yang berarti ia bisa dengan mudah diubah dari atau ke format seperti JSON. Properti utamanya adalah `results`, yang akan berisi daftar objek TvShow. Jadi, ketika aplikasi meminta data acara TV dari API, respons yang diterima akan langsung dimasukkan ke dalam objek TvShowResponse ini, dan daftar acara TV yang sebenarnya bisa diakses melalui properti `results`.

- **activity\_main.xml**

File `activity_main.xml` ini adalah tata letak utama (layout) untuk MainActivity di aplikasi. Secara sederhana, ia berfungsi sebagai bingkai kosong yang akan diisi oleh komponen UI yang lebih kecil, yaitu Fragment. Di dalamnya, ada `FrameLayout` dengan ID `fragment_container` yang ukurannya memenuhi seluruh layar. `FrameLayout` ini adalah tempat di mana berbagai Fragment (seperti `HomeFragment` atau `DetailFragment`) akan "diletakkan" dan ditampilkan secara dinamis oleh aplikasi. Jadi, `activity_main.xml` ini adalah fondasi visual yang akan menampung semua tampilan utama aplikasi.

- **fragment\_detail.xml**

File `fragment_detail.xml` ini adalah tata letak (layout) untuk tampilan detail sebuah acara TV yang akan ditampilkan di `DetailFragment`. Layout ini menggunakan Data Binding, ditandai dengan tag `<layout>` dan `<data>`, yang memungkinkan data dari objek TvShow langsung ditampilkan ke elemen-elemen UI tanpa perlu banyak kode di Java/Kotlin. Seluruh konten detail acara TV diletakkan di dalam `ScrollView` agar bisa digulir jika informasinya terlalu panjang. Di dalamnya, ada `ImageView` untuk menampilkan poster acara TV, `TextView` untuk menampilkan judul, tanggal tayang, dan rating, serta `ImageButton` untuk tombol favorit. Ada juga dua Button yaitu "Open Link" untuk membuka halaman web acara TV, dan "Back" untuk kembali ke tampilan sebelumnya. Semua elemen ini disusun secara vertikal dan dipusatkan agar mudah dilihat.

- **fragment\_favorite.xml**

File `fragment_favorite.xml` ini adalah tata letak (layout) untuk tampilan daftar acara TV favorit yang akan muncul di `FavoriteFragment`. Layout ini didesain dengan sebuah bilah judul (header bar) di bagian atas yang berisi tombol kembali dan judul "Favorit". Di bawah bilah judul tersebut, ada `RecyclerView` yang akan digunakan untuk menampilkan daftar semua acara TV yang sudah ditandai sebagai favorit dalam bentuk daftar yang bisa digulir. Jadi, `fragment_favorite.xml` ini bertugas untuk mengatur bagaimana tampilan daftar favorit tersebut akan terlihat di layar aplikasi.

- **fragment\_home.xml**

File `fragment_home.xml` ini adalah tata letak (layout) utama untuk tampilan beranda aplikasi yang akan muncul di `HomeFragment`. Bagian atas layout ini terdapat bilah judul (header bar) berwarna biru yang berisi judul aplikasi dan sebuah tombol. Tombol ini berfungsi untuk mengarahkan pengguna ke layar daftar acara TV favorit. Di bawah bilah judul, terdapat `RecyclerView` yang akan menampilkan daftar drama Tiongkok yang bisa digulir. Jadi, `fragment_home.xml` ini bertanggung jawab untuk mengatur bagaimana tampilan beranda aplikasi, lengkap dengan daftar drama dan tombol navigasi ke favorit, akan terlihat di layar.

- **item\_drama.xml**

File `item_drama.xml` ini adalah tata letak (layout) yang digunakan untuk menampilkan setiap satu item dalam daftar acara TV di `RecyclerView`. Setiap item ditampilkan di dalam sebuah `CardView` yang memberikan tampilan efek kartu dengan sudut membulat dan sedikit bayangan, membuatnya terlihat lebih modern. Layout ini menggunakan `Data Binding`, memungkinkan data dari objek `TvShow` langsung ditampilkan ke elemen UI yang relevan. Di dalam setiap kartu, terdapat `ImageView` untuk menampilkan poster acara TV, beberapa `TextView` untuk judul, tahun tayang, dan rating. Selain itu, ada dua tombol: "Watch Here" yang mengarahkan pengguna ke tautan eksternal, dan "Detail" yang akan membuka layar detail lengkap acara TV tersebut. Ini membuat setiap item daftar menjadi interaktif dan informatif.

## **Tautan Git**

Berikut adalah tautan untuk semua source code yang telah dibuat.

[firdakhrns/Pemrograman-Mobile](https://github.com/firdakhrns/Pemrograman-Mobile)