

**LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE**



Oleh:

Noviana Nur Aisyah NIM. 2310817120005

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025**

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Noviana Nur Aisyah
NIM : 2310817120005

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Zulfa Auliya Akbar
NIM. 2210817210026

Muti`a Maulida S.Kom M.T.I
NIP. 198810272019032013

DAFTAR ISI

DAFTAR GAMBAR

MODUL 1: ANDROID BASIC WITH KOTLIN

Gambar 1. Screenshot Hasil Jawaban Soal 1	12
Gambar 2. Screenshot Hasil Jawaban Soal 1 (2)	12

MODUL 2: ANDROID LAYOUT

Gambar 3. Screenshot Hasil Jawaban Soal 1	29
Gambar 4. Screenshot Hasil Jawaban Soal 1 (2)	29
Gambar 5. Screenshot Hasil Jawaban Soal 1 (3)	29
Gambar 6. Screenshot Hasil Jawaban Soal 1 (4)	30

MODUL 3: BUILD A SCROLLABLE LIST

Gambar 7. Screenshot Hasil Jawaban Soal 1	60
---	----

MODUL 4: VIEWMODEL & DEBUGGING

Gambar 8. Screenshot Hasil Jawaban Soal 1	98
Gambar 9. Screenshot Timber Button Soal 1 (2)	98

MODUL 5: CONNECT TO THE INTERNET

Gambar 10. Screenshot Hasil Soal 1	139
--	-----

DAFTAR TABEL

MODUL 1: ANDROID BASIC WITH KOTLIN

Tabel 1. Source Code MainActivity.kt Soal 1	8
Tabel 2. Source Code MainActivityViewModel.kt Soal 1	9
Tabel 3. Source Code activity_main.xml Soal 1	10
Tabel 4. Source Code tool_bar.xml Soal 1	11

MODUL 2: ANDROID LAYOUT

Tabel 5. Source Code MainActivity.kt Soal 1	18
Tabel 6. Source Code MainActivityViewModel.kt Soal 1	21
Tabel 7. Source Code activity_main.xml Soal 1	22

MODUL 3: BUILD A SCROLLABLE LIST

Tabel 8. Source Code MainActivity.kt Soal 1	35
Tabel 9. Source Code Song.kt Soal 1	37
Tabel 10. Source Code DetailFragment.kt Soal 1	37
Tabel 11. Source Code HomeFragment.kt Soal 1	39
Tabel 12. Source Code ListSongAdapter.kt Soal 1	43
Tabel 13. Source Code activity_main.xml Soal 1	45
Tabel 14. Source Code fragment_detail.xml Soal 1	46
Tabel 15. Source Code fragment_home.xml Soal 1	52
Tabel 16. Source Code item_song.xml Soal 1	53

MODUL 4: VIEWMODEL & DEBUGGING

Tabel 17. Source Code MainActivity.kt Soal 1	73
Tabel 18. Source Code Song.kt Soal 1	74
Tabel 19. Source Code DetailFragment.kt Soal 1	75
Tabel 20. Source Code HomeFragment.kt Soal 1	77
Tabel 21. Source Code HomeViewModel.kt Soal 1	81

Tabel 22. Source Code ListSongAdapter.kt Soal 1	84
Tabel 23. Source Code activity_main.xml Soal 1	86
Tabel 24. Source Code fragment_detail.xml Soal 1	86
Tabel 24. Source Code fragment_home.xml Soal 1	91
Tabel 25. Source Code item_song.xml Soal 1	93

MODUL 5: CONNECT TO THE INTERNET

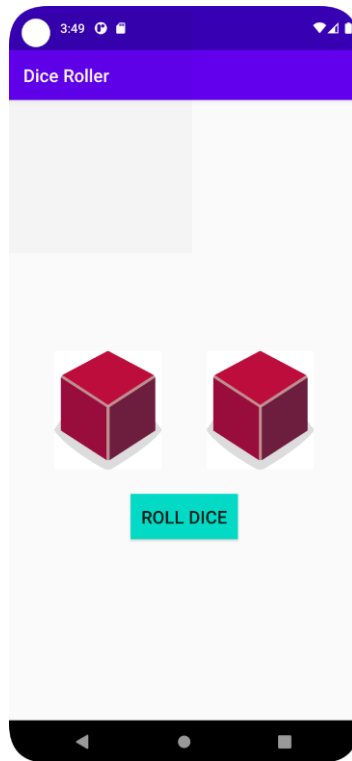
Tabel 26. Source Code MainActivity.kt Soal 1	113
Tabel 27. Source Code MovieDao.kt Soal 1	119
Tabel 28. Source Code AppDatabase.kt Soal 1	120
Tabel 29. Source Code MovieAppPreferences.kt Soal 1	120
Tabel 30. Source Code RetrofitClient.kt Soal 1	122
Tabel 31. Source Code TmdbApiService.kt Soal 1	123
Tabel 32. Source Code MovieDto.kt Soal 1	124
Tabel 33. Source Code MovieDtoExtension.kt Soal 1	126
Tabel 34. Source Code MovieListResponse.kt Soal 1	127
Tabel 35. Source Code MovieRepository.kt Soal 1	127
Tabel 36. Source Code Movie.kt Soal 1	130
Tabel 37. Source Code GetPopularMoviesUseCase.kt Soal 1	130
Tabel 38. Source Code DetailActivity.kt Soal 1	131
Tabel 39. Source Code MovieAdapter.kt Soal 1	134
Tabel 40. Source Code MovieViewModel.kt Soal 1	136
Tabel 41. Source Code ViewModelFactory.kt Soal 1	138
Tabel 42. Source Code Result Soal 1	139

MODUL 1: ANDROID BASIC WITH KOTLIN

SOAL 1

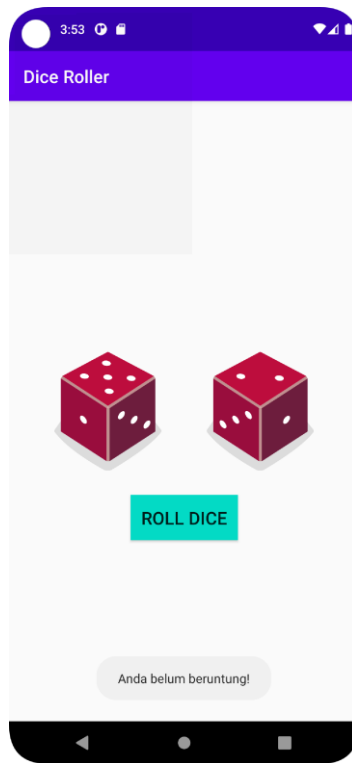
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



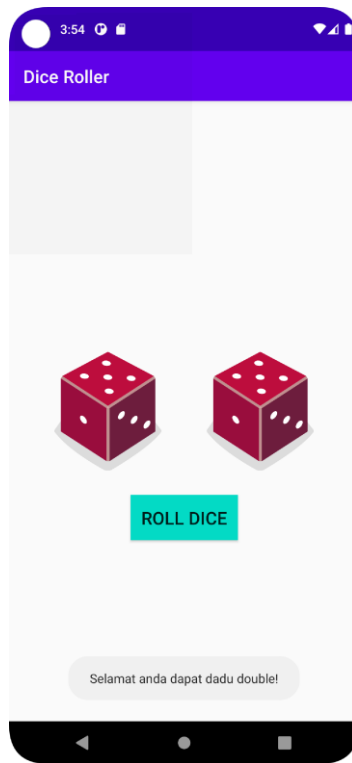
Gambar 1 Tampilan Awal Aplikasi

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.



Gambar 3 Tampilan Dadu Double

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

5. Untuk gambar dadu dapat didownload pada link berikut:

https://drive.google.com/file/d/14V3qXGdFnuoYN4AGd_9SgFh8kw8X9ySm/view

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code MainActivity.kt Soal 1

1	package	com.example.diceroller
2		
3	import	android.os.Bundle

4	import	android.widget.Button
5	import	android.widget.ImageView
6	import	android.widget.Toast
7	import	androidx.appcompat.app.AppCompatActivity
8		
9	class	MainActivity : AppCompatActivity() {
10	private	var firstDice: Int? = null
11	override	fun onCreate(savedInstanceState:
12	Bundle?)	{
13		super.onCreate(savedInstanceState)
14		setContentView(R.layout.activity_main)
15		
16	val	rollButton: Button =
17	findViewById(R.id.button)	
18		rollButton.setOnClickListener {
19		rollDice()
20		rollDice2()
21		}
22		}

2. MainActivityViewModel.kt

Tabel 2. Source Code MainActivityViewModel.kt Soal 1

1	package	com.example.diceroller
2		
3	import	android.os.Bundle
4	import	android.widget.Button
5	import	android.widget.ImageView
6	import	android.widget.Toast
7	import	androidx.appcompat.app.AppCompatActivity
8		

9	class MainActivity : AppCompatActivity() {
10	private var firstDice: Int? = null
11	override fun onCreate(savedInstanceState:
12	Bundle?) {
13	super.onCreate(savedInstanceState)
14	setContentView(R.layout.activity_main)
15	
16	val rollButton: Button =
17	findViewById(R.id.button)
18	rollButton.setOnClickListener {
19	rollDice()
20	rollDice2()
21	}
22	}

3. activity_main.xml

Tabel 3. Source Code activity_main.xml Soal 1

1	package com.example.diceroller
2	
3	import android.os.Bundle
4	import android.widget.Button
5	import android.widget.ImageView
6	import android.widget.Toast
7	import androidx.appcompat.app.AppCompatActivity
8	
9	class MainActivity : AppCompatActivity() {
10	private var firstDice: Int? = null
11	override fun onCreate(savedInstanceState:
12	Bundle?) {
13	super.onCreate(savedInstanceState)

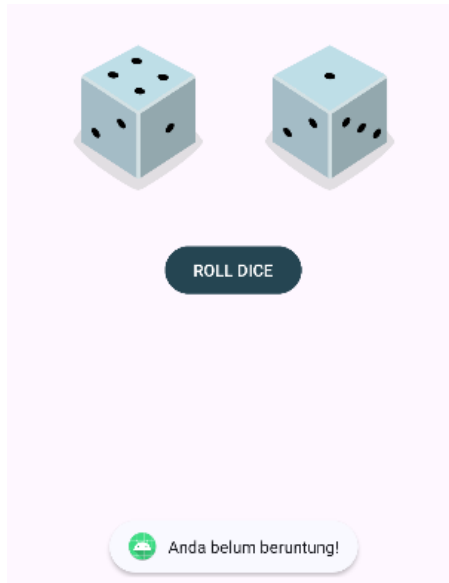
14	setContentView(R.layout.activity_main)
15	
16	val rollButton: Button =
17	findViewById(R.id.button)
18	rollButton.setOnClickListener {
19	rollDice()
20	rollDice2()
21	}
22	}

4. tool_bar.xml

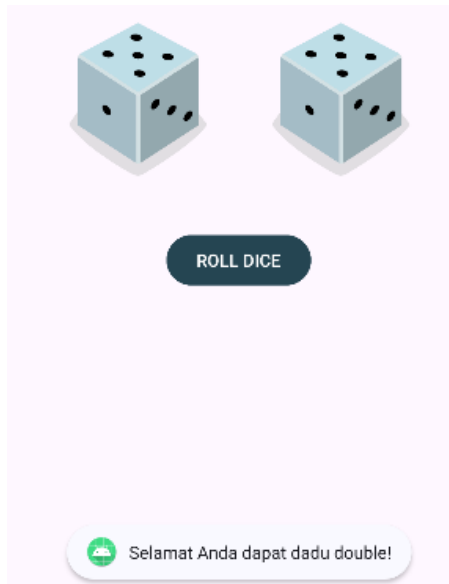
Tabel 4. Source Code tool_bar.xml Soal 1

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

B. Output Program



Gambar 2. Screenshot Hasil Jawaban Soal 1



Gambar 2. Screenshot Hasil Jawaban Soal 1 (2)

C. Pembahasan

1. MainActivity.kt

File ini merupakan logika utama dari aplikasi untuk menampilkan UI dan mengatur interaksi pengguna. Terdapat beberapa bagian penting, yaitu:

- a) View Binding

```
binding =
    ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
```

Digunakan untuk menghubungkan file `activity_main.xml` (`ActivityMainBinding`).

b) Set gambar Dadu (setiap membuka aplikasi)

```
binding.ivDice.setImageResource(R.drawable.dice_0)
binding.ivDice2.setImageResource(R.drawable.dice_0)
```

Digunakan untuk menampilkan gambar dadu kosong setiap kali aplikasi dibuka.

c) Event Button

```
binding.btnRoll.setOnClickListener {
    viewModel.rollDice()
}
```

Ketika button diklik, maka akan meminta `ViewModel` untuk menghasilkan dua angka acak dan update Live Data `dice1` dan `dice2`.

d) LiveData

```
viewModel.dice1.observe(this) { hasil ->
    binding.ivDice.setImageResource(hasil)
}
viewModel.dice2.observe(this) { hasil ->
    binding.ivDice2.setImageResource(hasil)
    checkIfDouble()
}
```

Jika terdapat perubahan nilai `dice1` atau `dice2`, gambar dadu akan ter-update sesuai hasilnya, kemudian hasil ini akan dicek apakah sama (`checkIfDouble`).

`viewModel.dice1` adalah `LiveData` dari `ViewModel`, bersifat *read-only*.

`.observe(this)` adalah cara Android menyambungkan `LiveData` ke komponen UI seperti `Activity`, `this` di sini berarti konteks `Activity` saat ini. `{ hasil -> ... }` merupakan fungsi yang dipanggil setiap nilai `dice1` berubah dan hasil adalah nilai baru dari `dice1`.

`Binding.ivDice.setImageResource(hasil)` adalah jika `dice1` berubah (misalnya menjadi `R.drawable.dice_4`), maka gambar di `ImageView` bernama `ivDice` akan langsung berubah ke gambar dadu 4.

e) Cek Double Dadu

```
private fun checkIfDouble() {
    val d1 = viewModel.dice1.value
    val d2 = viewModel.dice2.value

    if (d1 == d2) {
        Toast.makeText(this, "Selamat Anda dapat
dadu double!", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "Anda belum
beruntung!", Toast.LENGTH_SHORT).show()
    }
}
```

Digunakan untuk menampilkan toast jika dua dadu sama (misalnya 6 dengan 6), yang artinya “*double*”.

f) Menyesuaikan Padding Toolbar Terhadap Status Bar

```
val statusBarHeight =
resources.getDimensionPixelSize(
    resources.getIdentifier("status_bar_height",
"dimen", "android")
)
```

```
val toolbar: Toolbar = findViewById(R.id.toolBar)
toolbar.setPadding(0, statusBarHeight, 0, 0)
```

Digunakan untuk mengatur padding atas toolbar agar tidak tumpang tindih dengan status bar perangkat.

2. MainActivityViewModel.kt

File ini memuat logika untuk generate dadu random dan mengatur LiveData.

```
private val _dice1 = MutableLiveData<Int>()
val dice1: LiveData<Int> get() = _dice1
```

Bagian ini digunakan untuk membedakan siapa yang boleh mengubah data dan siapa yang hanya dapat mengamati data.

`MutableLiveData<Int>()` artinya dapat bisa diubah, namun hanya dapat diakses dan diubah dari dalam `MainActivityViewMode.kt` karena `private`.

`val dice1: LiveData<Int> get() = _dice1` artinya adalah *read-only* dari versi `_dice1`. Tipe `LiveData` artinya tidak dapat diubah dari luar `MainActivityViewModel.kt`, namun dapat diamati/dibaca oleh `MainActivity.kt` (`viewModel.dice1.observe(this)`).

```
fun rollDice() {
    val dice = listOf(
        R.drawable.dice_1,
        R.drawable.dice_2,
        R.drawable.dice_3,
        R.drawable.dice_4,
        R.drawable.dice_5,
        R.drawable.dice_6
    )
    _dice1.value = dice.random()
    _dice2.value = dice.random()
}
```

Saat fungsi `rollDice()` dipanggil, kedua gambar dadu akan dipilih secara acak dan meng-update LiveData nya. UI akan secara otomatis ter-update karena observer di `MainActivity.kt`.

3. `activity_main.xml`

`<ConstraintLayout>` digunakan untuk layout utama.

Merupakan layout fleksibel untuk mengatur posisi komponen berdasarkan constraints (posisi relatif).

`<include .../>` digunakan untuk menyisipkan layout toolbar.

Digunakan untuk layout `tool_bar.xml` agar digunakan di sini, seperti *copy-paste* isi layout toolbar ke `activity_main.xml`.

`<ImageView/>` digunakan untuk gambar dadu 1.

`ivDice` untuk menampilkan gambar dadu pertama.

`<ImageView/>` digunakan untuk gambar dadu 2.

`ivDice2` digunakan untuk menampilkan gambar dadu kedua.

`<Button/>` digunakan untuk tombol roll dadu.

Merupakan tombol untuk rolling dadu, di mana ketika diklik akan memanggil fungsi `rollDice()` di `ViewModel`.

4. `tool_bar.xml`

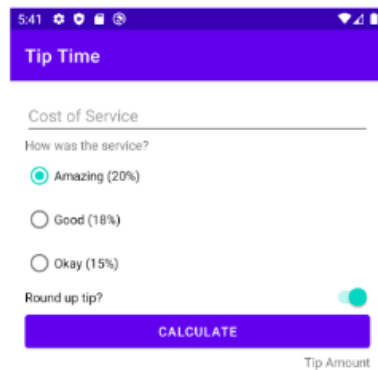
File ini merupakan layout khusus toolbar yang akan dimasukkan ke layout utama menggunakan `<include>`.

MODUL 2: ANDROID LAYOUT

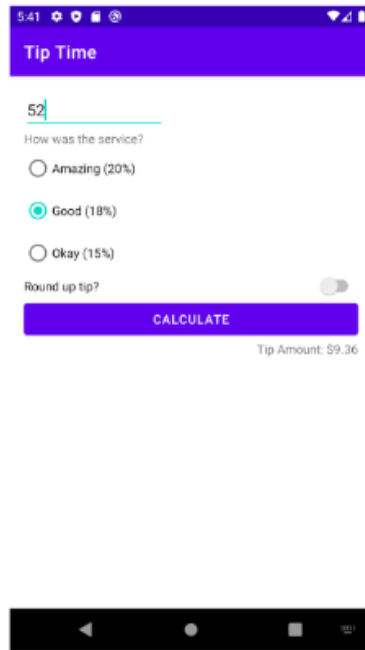
SOAL 1

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

Tabel 5. Source Code MainActivity.kt Soal 1

1	package com.example.calculatortip
2	
3	import android.os.Bundle
4	import androidx.activity.enableEdgeToEdge
5	import androidx.appcompat.app.AppCompatActivity
6	import androidx.core.view.ViewCompat
7	import androidx.core.view.WindowInsetsCompat
8	import androidx.lifecycle.ViewModelProvider
9	import androidx.lifecycle.get
10	import
11	com.example.calculatortip.databinding.ActivityMainBi
12	nding

```

13 import android.view.View
14 import android.os.Handler
15 import android.os.Looper
16 import android.widget.Toast
17 import
18 androidx.core.splashscreen.SplashScreen.Companion.in
19 stallSplashScreen
20
21 class MainActivity : AppCompatActivity() {
22     private lateinit var binding: ActivityMainBinding
23
24
25     override fun onCreate(savedInstanceState:
26 Bundle?) {
27         super.onCreate(savedInstanceState)
28
29         Thread.sleep(2000)
30         installSplashScreen()
31
32         binding =
33 ActivityMainBinding.inflate(layoutInflater)
34         setContentView(binding.root)
35
36         var viewModel =
37 ViewModelProvider(this).get(MainActivityViewModel::c
38 lass.java)
39
40         viewModel.tipResult.observe(this) { tip ->
41             binding.tipAmount.text = "Tip Amount:
42 $tip"

```

```

43         binding.tipAmount.visibility           =
44 View.VISIBLE
45     }
46
47     binding.calculateButton.setOnClickListener {
48         val             inputEditText           =
49 binding.inputText.text.toString()
50         val             radioGroup              =
51 binding.radioGroup.checkedRadioButtonId
52         val             switchRound             =
53 binding.switchRound.isChecked
54
55         val             inputValue              =
56 inputEditText.toDoubleOrNull()
57         if (inputValue == null) {
58             Toast.makeText(this, "Mohon isi Cost
59 Of Value dengan angka", Toast.LENGTH_SHORT).show()
60             binding.tipAmount.text = ""
61             binding.tipAmount.visibility       =
62 View.GONE
63             return@setOnClickListener
64         } else if (inputValue == 0.0) {
65             Toast.makeText(this, "Mohon isi
66 dengan angka yang lebih besar",
67 Toast.LENGTH_SHORT).show()
68             binding.tipAmount.text = ""
69             binding.tipAmount.visibility       =
70 View.GONE
71             return@setOnClickListener
72         }

```

73	
74	viewModel.calculate(inputEditText,
75	radioGroup, switchRound)
76	}
77	
78	}
79	}
80	

2. MainActivityViewModel.kt

Tabel 6. Source Code MainActivityViewModel.kt Soal 1

1	package	com.example.calculatortip
2		
3	import	androidx.lifecycle.ViewModel
4	import	androidx.lifecycle.LiveData
5	import	androidx.lifecycle.MutableLiveData
6	import	
7	com.google.android.material.textfield.TextInputEditT	
8	ext	
9	import	java.text.NumberFormat
10	import	kotlin.math.ceil
11		
12	class MainActivityViewModel:	ViewModel() {
13	private val	_tipResult =
14	MutableLiveData<String>()	
15	val tipResult: LiveData<String>	= _tipResult
16		
17	fun calculate(inputEditText: String?, radioGroup:	
18	Int, switchRound: Boolean)	{
19	val cost = inputEditText?.toDoubleOrNull()	

20	if (cost == null) {
21	_tipResult.value = ""
22	return
23	}
24	
25	val tipPercentage = when (radioGroup) {
26	R.id.amazing_button -> 0.20
27	R.id.good_button -> 0.18
28	else -> 0.15
29	}
30	
31	var tip = tipPercentage * cost
32	
33	if (switchRound) {
34	tip = ceil(tip)
35	}
36	
37	val formattedTip =
38	NumberFormat.getCurrencyInstance().format(tip)
39	
40	_tipResult.value = formattedTip
41	}
42	
43	
44	}

3. activity_main.xml

Tabel 7. Source Code activity_main.xml Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	

3	<androidx.constraintlayout.widget.ConstraintLayout
4	xmlns:android="http://schemas.android.com/apk/res/a
5	ndroid"
6	xmlns:app="http://schemas.android.com/apk/res-
7	auto"
8	xmlns:tools="http://schemas.android.com/tools"
9	android:id="@+id/main"
10	android:layout_width="match_parent"
11	android:layout_height="match_parent"
12	tools:context=".MainActivity">
13	
14	<TextView
15	android:id="@+id/toolBarText"
16	android:layout_width="match_parent"
17	android:layout_height="wrap_content"
18	android:background="@android:color/white"
19	android:backgroundTint="#547792"
20	android:fontFamily="sans-serif"
21	android:paddingTop="20dp"
22	android:paddingBottom="15dp"
23	android:paddingLeft="15dp"
24	android:paddingRight="15dp"
25	android:text="Tip Time"
26	android:textColor="@color/white"
27	android:textSize="20sp"
28	android:textStyle="bold"
29	app:layout_constraintEnd_toEndOf="parent"
30	app:layout_constraintHorizontal_bias="0.5"
31	
32	app:layout_constraintStart_toStartOf="parent"

33	/>
34	
35	<EditText
36	android:id="@+id/input_text"
37	android:layout_width="380dp"
38	android:layout_height="wrap_content"
39	android:layout_marginTop="10dp"
40	android:inputType="text"
41	app:layout_constraintEnd_toEndOf="parent"
42	app:layout_constraintHorizontal_bias="0.5"
43	
44	app:layout_constraintStart_toStartOf="parent"
45	
46	app:layout_constraintTop_toBottomOf="@id/toolBarText"
47	
48	android:hint="Cost of Service"
49	android:textColorHint="#aaa7ad"
50	/>
51	
52	<TextView
53	android:id="@+id/pertanyaan"
54	android:layout_width="wrap_content"
55	android:layout_height="wrap_content"
56	android:text="How was the service? "
57	android:textColor="#808080"
58	app:layout_constraintHorizontal_bias="0.06"
59	app:layout_constraintEnd_toEndOf="parent"
60	
61	app:layout_constraintStart_toStartOf="parent"
62	


```

63
64 app:layout_constraintTop_toBottomOf="@id/input_text
65 " />
66
67     <RadioGroup
68         android:id="@+id/radio_group"
69         android:layout_width="wrap_content"
70         android:layout_height="wrap_content"
71         android:orientation="vertical"
72
73 app:layout_constraintTop_toBottomOf="@id/pertanyaan
74 "
75
76 app:layout_constraintStart_toStartOf="parent"
77     app:layout_constraintEnd_toEndOf="parent"
78     app:layout_constraintHorizontal_bias="0.06"
79     android:layout_marginTop="5dp"
80 >
81
82     <RadioButton
83         android:id="@+id/amazing_button"
84         android:layout_width="wrap_content"
85         android:layout_height="wrap_content"
86         android:text="Amazing (20%)"
87
88 app:buttonTint="@drawable/radio_button_selector"/>
89
90     <RadioButton
91         android:id="@+id/good_button"
92         android:layout_width="wrap_content"

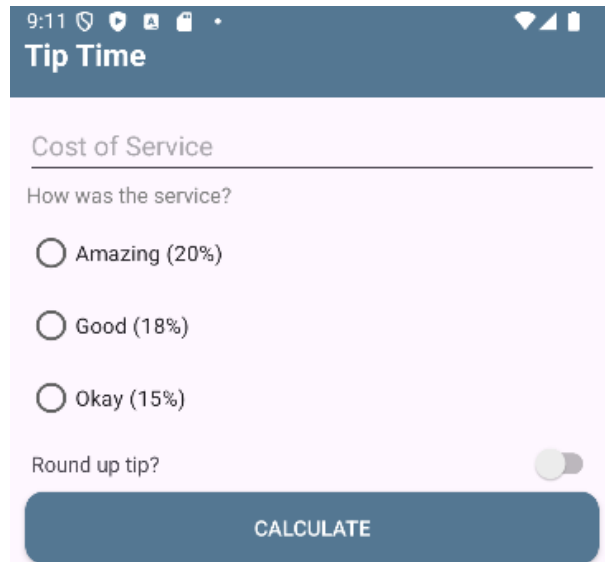
```

93	android:layout_height="wrap_content"
94	android:text="Good (18%)"
95	
96	app:buttonTint="@drawable/radio_button_selector"/>
97	
98	<RadioButton
99	android:id="@+id/okay_button"
100	android:layout_width="wrap_content"
101	android:layout_height="wrap_content"
102	android:text="Okay (15%)"
103	
104	app:buttonTint="@drawable/radio_button_selector"/>
105	</RadioGroup>
106	
107	<TextView
108	android:id="@+id/round_up"
109	android:layout_width="wrap_content"
110	android:layout_height="wrap_content"
111	android:text="Round up tip? "
112	android:layout_marginTop="10dp"
113	app:layout_constraintHorizontal_bias="0.06"
114	app:layout_constraintEnd_toEndOf="parent"
115	
116	app:layout_constraintStart_toStartOf="parent"
117	
118	app:layout_constraintTop_toBottomOf="@id/radio_grou
119	p"
120	/>
121	
122	<Switch

123	android:id="@+id/switch_round"
124	android:layout_width="wrap_content"
125	android:layout_height="wrap_content"
126	android:layout_marginEnd="15dp"
127	
128	app:layout_constraintBottom_toBottomOf="@id/round_u
129	p"
130	app:layout_constraintEnd_toEndOf="parent"
131	
132	app:layout_constraintTop_toTopOf="@id/round_up"
133	app:thumbTint="#eeeeee"
134	app:trackTint="#547792"
135	/>
136	
137	<android.widget.Button
138	android:id="@+id/calculate_button"
139	android:layout_width="380dp"
140	android:layout_height="wrap_content"
141	
142	android:background="@drawable/rounded_border"
143	app:layout_constraintEnd_toEndOf="parent"
144	
145	app:layout_constraintStart_toStartOf="parent"
146	
147	app:layout_constraintTop_toBottomOf="@id/switch_rou
148	nd"
149	android:text="Calculate"
150	android:layout_margin="5dp"
151	android:textColor="@color/white"
152	/>

```
153
154     <TextView
155         android:id="@+id/tip_amount"
156         android:layout_width="wrap_content"
157         android:layout_height="wrap_content"
158         android:text="apa saja brok "
159         android:layout_marginTop="5dp"
160         app:layout_constraintHorizontal_bias="0.94"
161         app:layout_constraintEnd_toEndOf="parent"
162
163     app:layout_constraintStart_toStartOf="parent"
164
165     app:layout_constraintTop_toBottomOf="@id/calculate_
166     button"
167         android:visibility="gone"
168     />
169
170 </androidx.constraintlayout.widget.ConstraintLayout
171 >
```

B. Output Program



9:11

Tip Time

Cost of Service

How was the service?

☐ Amazing (20%)

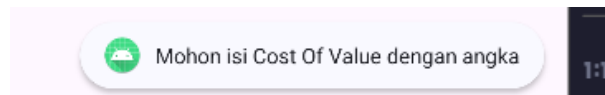
☐ Good (18%)

☐ Okay (15%)

Round up tip? ☐

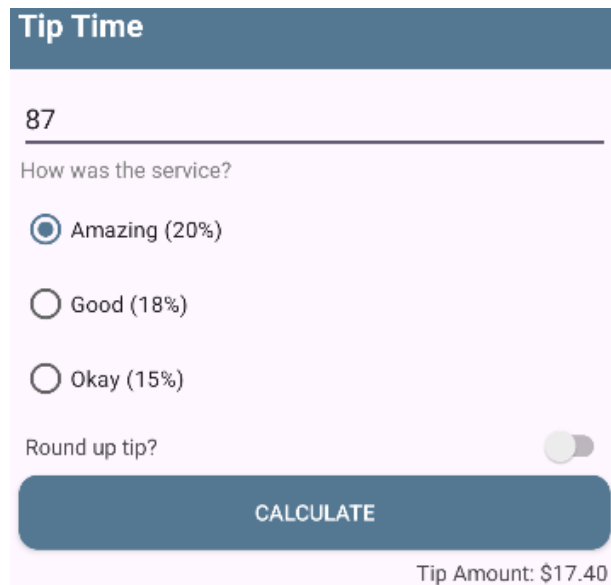
CALCULATE

Gambar 3. Screenshot Hasil Jawaban Soal 1



Mohon isi Cost Of Value dengan angka

Gambar 4. Screenshot Hasil Jawaban Soal 1 (2)



Tip Time

87

How was the service?

☒ Amazing (20%)

☐ Good (18%)

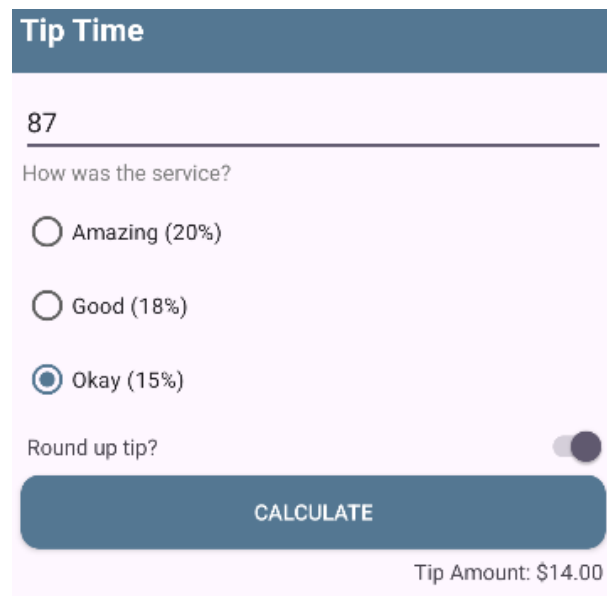
☐ Okay (15%)

Round up tip? ☐

CALCULATE

Tip Amount: \$17.40

Gambar 5. Screenshot Hasil Jawaban Soal 1 (3)



Gambar 6. Screenshot Hasil Jawaban Soal 1 (4)

C. Pembahasan

1. MainActivity.kt

File ini merupakan logika utama dari aplikasi untuk menampilkan UI dan mengatur interaksi pengguna. Terdapat beberapa bagian penting, yaitu:

g) Splash Screen

```
Thread.sleep(2000)
installSplashScreen()
```

Digunakan untuk menampilkan splash screen saat aplikasi pertama kali dibuka. `Thread.sleep(2000)` artinya program memberi delay selama 2 detik agar splash terlihat cukup lama, lalu `installSplashScreen()` menampilkan splash screen dari android.

h) View Binding

```
binding =
    ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
```

Digunakan untuk menghubungkan layout `activity_main.xml` ke `MainActivity` agar elemen UI dapat diakses langsung melalui `binding`.

i) ViewModel & LiveData

```
var viewModel =
    ViewModelProvider(this).get(MainActivityViewModel
        ::class.java)

viewModel.tipResult.observe(this) { tip ->
    binding.tipAmount.text = "Tip Amount: $tip"
    binding.tipAmount.visibility = View.VISIBLE
}
```

ViewModelProvider digunakan untuk menghubungkan MainActivity ke MainActivityViewModel. Kemudian, tipResult merupakan LiveData berisi hasil kalkulasi. Saat nilai tipResult berubah, tipAmount akan menampilkan hasilnya.

j) Event Button

```
binding.calculateButton.setOnClickListener {
    ...
    viewModel.calculate(inputEditText, radioGroup,
        switchRound)
}
```

Ketika button calculateButton diklik, maka akan:

- Mengambil input nilai biaya (inputEditText);
- Mengecek radio button yang dipilih;
- Mengecek apakah pembulatan / round up aktif (switch);
- Jika input kosong atau 0, akan muncul toast sebagai peringatan;
- Jika valid, memanggil fungsi calculate() dari View Model untuk memprosesnya.

k) Validasi Input

```
val inputValue = inputEditText.toDoubleOrNull()
if (inputValue == null || inputValue == 0.0) {
    Toast.makeText(this, ...).show()
}
```

```

        binding.tipAmount.text = ""
        binding.tipAmount.visibility = View.GONE
        return@setOnClickListener
    }

```

Jika input tidak valid, maka akan muncul peringatan berupa toast, kemudian teks hasil tip disembunyikan.

l) Tampilan Hasil Tip

```

binding.tipAmount.text = "Tip Amount: $tip"
binding.tipAmount.visibility = View.VISIBLE

```

Jika kalkulasi berhasil, maka hasil tip akan ditampilkan di textView tipAmount.

2. MainActivityViewModel.kt

File ini memuat ViewModel dari aplikasi kalkulator tip untuk memproses logika perhitungan tip berdasarkan input pengguna. View Model ini akan memberikan hasil ke UI melalui LiveData. Terdapat beberapa bagian, yaitu:

a) LiveData dan MutableLiveData

```

private val _tipResult = MutableLiveData<String>()
val tipResult: LiveData<String> = _tipResult

```

Fungsi dari _tipResult adalah untuk menyimpan nilai hasil kalkulasi tip dan bisa diubah dari dalam ViewModel. Kemudian, tipResult adalah versi read-only dari _tipResult yang digunakan oleh Activity untuk mengamati perubahan dan menampilkannya di UI.

b) Fungsi calculate()

```

fun calculate(inputEditText: String?, radioGroup:
Int, switchRound: Boolean)

```

Fungsi ini dapat menerima tiga parameter dari UI, yaitu inputEditText, radioGroup, dan switchRound.

c) Validasi Input

```

val cost = inputEditText?.toDoubleOrNull()

```



```

if (cost == null) {
    _tipResult.value = ""
    return
}

```

Jika input kosong atau bukan merupakan angka, maka hasil tidak akan ditampilkan.

d) Menentukan Persentase Tip

```

val tipPercentage = when (radioGroup) {
    R.id.amazing_button -> 0.20
    R.id.good_button -> 0.18
    else -> 0.15
}

```

Bagian ini berfungsi untuk menentukan persentase tip berdasarkan pilihan radio button.

e) Perhitungan dan Pembulatan

```

var tip = tipPercentage * cost
if (switchRound) {
    tip = ceil(tip)
}

```

Bagian ini berfungsi untuk mengalikan nilai cost dengan tipPercentage. Jika opsi pembulatan (switchRound) aktif, maka hasil dibulatkan ke atas dengan `ceil()`.

f) Format Mata Uang dan Update UI

```

val formattedTip =
    NumberFormat.getCurrencyInstance().format(tip)
_tipResult.value = formattedTip

```

Hasil tip diformat ke format uang lokal (misalnya "Rp12.000,00" atau "\$5.00"), lalu diberikan ke `_tipResult`, yang akan langsung dikirim ke UI melalui observer di Activity.

3. activity_main.xml

a) ConstraintLayout

Digunakan untuk menatur posisi antar elemen secara efisien dan responsive.

b) textView – toolBarText

Digunakan sebagai header aplikasi yang menampilkan judul, di mana bagian ini diberi warna background, padding dan teks bold agar terlihat seperti toolbar.

c) EditText – input_text

Digunakan untuk memasukkan tip.

d) TextView (pertanyaan)

Berisi pertanyaan kepada pengguna tentang seberapa bagus layanan yang diterima.

e) RadioGroup + RadioButton

Merupakan pilihan tingkat layanan, yaitu Amazing, Good dan Okay. Digunakan untuk menentukan persentasi tip.

f) TextView + Switch – Round Up

Digunakan untuk membulatkan hasil tip ke atas. Switch digunakan agar pengguna dapat mengaktifkan atau menonaktifkan fitur ini.

g) Button – Calculate

Digunakan untuk menghitung tip berdasarkan input dan pilihan pengguna.

h) TextView – tip_amount

Digunakan untuk menampilkan hasil tip yang dihitung. Pada awalnya tidak terlihat, namun akan ditampilkan setelah tombol ditekan dan hasil tersedia.

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.
3. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius.
 4. 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.
 5. Aplikasi menggunakan arsitektur single activity (satu activity memiliki beberapa fragment).
 6. Aplikasi berbasis XML harus menggunakan ViewBinding.

A. Source Code

1. MainActivity.kt

Tabel 8. Source Code MainActivity.kt Soal 1

```
1 package com.example.scrollablelist
2
3 import android.os.Bundle
```

```

4      import                                android.util.Log
5      import                                androidx.activity.enableEdgeToEdge
6      import                                androidx.appcompat.app.AppCompatActivity
7      import                                androidx.core.view.ViewCompat
8      import                                androidx.core.view.WindowInsetsCompat
9      import
10     com.example.scrollablelist.databinding.ActivityMainB
11     inding
12
13     class MainActivity : AppCompatActivity() {
14         private lateinit var binding: ActivityMainBinding
15
16         override fun onCreate(savedInstanceState:
17 Bundle?) {
18             super.onCreate(savedInstanceState)
19
20             binding =
21 ActivityMainBinding.inflate(layoutInflater)
22             setContentView(binding.root)
23
24             val fragmentManager = supportFragmentManager
25             val homeFragment = HomeFragment()
26             val fragment =
27 fragmentManager.findFragmentByTag(HomeFragment::clas
28 s.java.simpleName)
29             if (fragment !is HomeFragment) {
30                 Log.d("MyFlexibleFragment", "Fragment
31 Name : " + HomeFragment::class.java.simpleName)
32                 fragmentManager
33                     .beginTransaction()

```

```

34         .add(R.id.main, homeFragment,
35     HomeFragment::class.java.simpleName)
36         .commit()
37     }
38 }
39 }
40

```

2. Song.kt

Tabel 9. Source Code Song.kt Soal 1

1	package com.example.scrollablelist
2	
3	import android.icu.text.CaseMap.Title
4	import android.os.Parcelable
5	import kotlinx.parcelize.Parcelize
6	
7	@Parcelize
8	data class Song(
9	val title: String,
10	val link: String,
11	val photo: String,
12	val singer: String,
13	val album: String,
14	val lyrics: String,
15	val desc: String
16	
17):Parcelable

3. DetailFragment.kt

Tabel 10. Source Code DetailFragment.kt Soal 1

```

1 package com.example.scrollablelist

```

```

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 com.bumptech.glide.Glide
9  import
10 com.example.scrollablelist.databinding.FragmentDetailBinding
11
12 class DetailFragment : Fragment() {
13
14     private var _binding: FragmentDetailBinding? = null
15     private val binding get() = _binding!!
16
17     override fun onCreateView(
18         inflater: LayoutInflater,
19         container: ViewGroup?,
20         savedInstanceState: Bundle?
21     ): View? {
22         _binding = FragmentDetailBinding.inflate(inflater,
23 container, false)
24
25         val title = arguments?.getString("TITLE")
26         val photo = arguments?.getString("PHOTO")
27         val singer = arguments?.getString("SINGER")
28         val album = arguments?.getString("ALBUM")
29         val lyrics = arguments?.getString("LYRICS")
30
31         binding.songTitle.text = title

```

```

32         photo?.let {
33             Glide.with(requireContext())
34                 .load(it)
35                 .into(binding.songCover)
36         }
37         binding.songSinger.text = singer
38         binding.songAlbum.text = album
39         binding.songLyrics.text = lyrics
40
41         binding.toolbar.setNavigationOnClickListener {
42             parentFragmentManager.popBackStack()
43         }
44
45         return binding.root
46     }
47
48     override fun onDestroyView() {
49         super.onDestroyView()
50         _binding = null
51     }
52 }

```

4. HomeFragment.kt

Tabel 11. Source Code HomeFragment.kt Soal 1

```

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

```

```

7   import
8   androidx.constraintlayout.helper.widget.Carousel.Ada
9   pter
10  import androidx.fragment.app.Fragment
11  import
12  com.example.scrollablelist.databinding.FragmentHomeB
13  inding
14  import android.content.Intent
15  import android.net.Uri
16  import androidx.core.os.BundleOf
17  import
18  androidx.recyclerview.widget.LinearLayoutManager
19
20  class HomeFragment : Fragment() {
21      private var _binding: FragmentHomeBinding? = null
22      private val binding get() = _binding!!
23
24      private lateinit var songAdapter: ListSongAdapter
25      private val list = ArrayList<Song>()
26
27      override fun onCreateView(
28          inflater: LayoutInflater,
29          container: ViewGroup?,
30          savedInstanceState: Bundle?
31      ): View? {
32          _binding =
33          FragmentHomeBinding.inflate(inflater, container,
34          false)
35
36          list.clear()

```



```

37         list.addAll(getListSong())
38         setupRecyclerView()
39
40         return binding.root
41     }
42
43     private fun setupRecyclerView() {
44         songAdapter = ListSongAdapter(
45             list,
46             onSpotifyClick = { link ->
47                 val intent =
48                 Intent(Intent.ACTION_VIEW, Uri.parse(link))
49                 startActivity(intent)},
50             onDetailClick = { title, photo, singer,
51 album, lyrics ->
52                 val detailFragment =
53                 DetailFragment().apply {
54                     arguments = Bundle().apply {
55                         putString("TITLE", title)
56                         putString("PHOTO", photo)
57                         putString("SINGER", singer)
58                         putString("ALBUM", album)
59                         putString("LYRICS", lyrics)
60                     }
61                 }
62
63
64         parentFragmentManager.beginTransaction()
65             .replace(R.id.main,
66                 detailFragment)

```

```

67         .addToBackStack(null)
68         .commit()
69     }
70 )
71
72     binding.rvSong.apply {
73         layoutManager =
74         LinearLayoutManager(context)
75         adapter = songAdapter
76         setHasFixedSize(true)
77     }
78 }
79
80     private fun getListSong() : ArrayList<Song> {
81         val          dataTitle          =
82         resources.getStringArray(R.array.data_title)
83         val          dataLink           =
84         resources.getStringArray(R.array.data_link)
85         val          dataPhoto          =
86         resources.getStringArray(R.array.data_photo)
87         val          dataSinger         =
88         resources.getStringArray(R.array.data_singer)
89         val          dataAlbum          =
90         resources.getStringArray(R.array.data_album)
91         val          dataLyrics         =
92         resources.getStringArray(R.array.data_lyrics)
93         val          dataDesc           =
94         resources.getStringArray(R.array.data_desc)
95         val listSong = ArrayList<Song>()
96

```

```

97         for (i in dataTitle.indices) {
98             val song = Song(title = dataTitle[i], link
99 = dataLink[i], photo = dataPhoto[i], singer =
100 dataSinger[i], album = dataAlbum[i], lyrics =
101 dataLyrics[i], desc = dataDesc[i])
102             listSong.add(song)
103         }
104
105         return listSong
106     }
107
108     override fun onDestroyView() {
109         super.onDestroyView()
110         _binding = null
111     }
112 }
113
114
115
116

```

5. ListSongAdapter.kt

Tabel 12. Source Code ListSongAdapter.kt Soal 1

```

1 package com.example.scrollablelist
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6 import com.bumptech.glide.Glide
7

```

```

8  import
9  com.example.scrollablelist.databinding.ItemSongBindi
10 ng
11
12 class ListSongAdapter(
13     private val listSong: ArrayList<Song>,
14     private val onSpotifyClick: (String) -> Unit,
15     private val onDetailClick: (String, String,
16 String, String, String) -> Unit)
17     :
18 RecyclerView.Adapter<ListSongAdapter.ListViewHolder>
19 () {
20
21     inner class ListViewHolder(val binding:
22 ItemSongBinding) :
23 RecyclerView.ViewHolder(binding.root) {
24         fun bind(song: Song) {
25             binding.tvTitle.text = song.title
26             binding.tvDesc.text = song.desc
27             Glide.with(binding.root.context)
28                 .load(song.photo)
29                 .into(binding.imgCover)
30
31
32             binding.buttonSpotify.setOnClickListener
33 {
34             onSpotifyClick(song.link)
35         }
36
37

```

```

38             binding.buttonDetail.setOnClickListener
39 {
40             onClick(song.title,
41 song.photo, song.singer, song.album, song.lyrics)
42         }
43     }
44 }
45
46     override fun onCreateViewHolder(parent:
47 ViewGroup, viewType: Int): ListViewHolder {
48         val binding =
49 ItemSongBinding.inflate(LayoutInflater.from(parent.c
50 ontext), parent, false)
51         return ListViewHolder(binding)
52     }
53
54     override fun getItemCount(): Int {
55         return listSong.size
56     }
57
58     override fun onBindViewHolder(holder:
59 ListViewHolder, position: Int) {
60         holder.bind(listSong[position])
61     }
62 }

```

6. activity_main.xml

Tabel 13. Source Code activity_main.xml Soal 1

```

1  <?xml version="1.0" encoding="utf-8"?>
2

```

```

3 <FrameLayout
4   xmlns:android="http://schemas.android.com/apk/res/andr
5   oid"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:id="@+id/main"
8       android:layout_width="match_parent"
9       android:layout_height="match_parent"
1      tools:context=".MainActivity">
0 </FrameLayout>

```

7. fragment_detail.xml

Tabel 14. Source Code fragment_detail.xml Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	xmlns:android="http://schemas.android.com/apk/res/a
4	ndroid"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	xmlns:app="http://schemas.android.com/apk/res-
8	auto"
9	xmlns:tools="http://schemas.android.com/tools"
10	tools:context=".DetailFragment">
11	
12	<androidx.appcompat.widget.Toolbar
13	android:id="@+id/toolbar"
14	android:layout_width="match_parent"
15	android:layout_height="wrap_content"
16	
17	android:background="@drawable/linear_gradient"
18	android:theme="?attr/actionBarTheme"

```

19         app:title="About Song"
20         app:titleTextColor="@color/white"
21         app:navigationIcon="@drawable/arrow_back"
22         app:layout_constraintTop_toTopOf="parent"
23
24     app:layout_constraintStart_toStartOf="parent"
25         app:layout_constraintEnd_toEndOf="parent"
26     />
27
28     <androidx.core.widget.NestedScrollView
29         android:id="@+id/scroll_view"
30         android:layout_width="0dp"
31         android:layout_height="0dp"
32
33     app:layout_constraintTop_toBottomOf="@id/toolbar"
34
35     app:layout_constraintBottom_toBottomOf="parent"
36
37     app:layout_constraintStart_toStartOf="parent"
38         app:layout_constraintEnd_toEndOf="parent"
39         android:scrollbars="vertical"
40         android:padding="16dp"
41         android:layout_margin="8dp"
42     >
43
44     <LinearLayout
45         android:layout_width="match_parent"
46         android:layout_height="wrap_content"
47         android:orientation="vertical"
48         android:padding="8dp">

```

49	
50	<ImageView
51	android:id="@+id/song_cover"
52	android:layout_width="300dp"
53	android:layout_height="300dp"
54	android:layout_marginTop="8dp"
55	
56	android:src="@drawable/place_holder"
57	
58	android:layout_gravity="center_horizontal"
59	/>
60	
61	<TextView
62	android:id="@+id/song_title"
63	android:layout_width="wrap_content"
64	
65	android:layout_height="wrap_content"
66	
67	app:layout_constraintStart_toStartOf="parent"
68	
69	app:layout_constraintEnd_toEndOf="parent"
70	
71	app:layout_constraintTop_toBottomOf="@id/song_cover
72	"
73	android:layout_marginTop="16dp"
74	android:text="TEDDY PICKER"
75	android:textStyle="bold"
76	android:textSize="20sp"
77	android:padding="2dp"
78	

79	
80	android:layout_gravity="center_horizontal"
81	/>
82	
83	<TextView
84	android:id="@+id/song_singer"
85	android:layout_width="wrap_content"
86	
87	android:layout_height="wrap_content"
88	
89	app:layout_constraintStart_toStartOf="parent"
90	
91	app:layout_constraintEnd_toEndOf="parent"
92	
93	app:layout_constraintTop_toBottomOf="@id/song_title
94	"
95	android:layout_marginTop="16dp"
96	android:padding="2dp"
97	android:text="Singer: Arctic
98	Monkeys"
99	android:textSize="14sp"
100	
101	android:layout_gravity="center_horizontal"
102	/>
103	
104	<TextView
105	android:id="@+id/song_album"
106	android:layout_width="wrap_content"
107	
108	android:layout_height="wrap_content"

```

109
110 app:layout_constraintStart_toStartOf="parent"
111
112 app:layout_constraintEnd_toEndOf="parent"
113         android:layout_marginTop="4dp"
114         android:padding="2dp"
115
116 app:layout_constraintTop_toBottomOf="@id/song_singer"
117
118         android:text="Album: Favourite
119 Worst Nightmare"
120         android:textSize="14sp"
121
122 android:layout_gravity="center_horizontal"
123     />
124
125     <TextView
126         android:id="@+id/song_content"
127         android:layout_width="wrap_content"
128
129 android:layout_height="wrap_content"
130
131 app:layout_constraintStart_toStartOf="parent"
132
133 app:layout_constraintEnd_toEndOf="parent"
134         android:layout_marginTop="16dp"
135         android:padding="2dp"
136
137 app:layout_constraintTop_toBottomOf="@id/song_album
138 "

```

139	android:text="Lyrics"
140	android:textStyle="bold"
141	android:textSize="14sp"
142	
143	android:layout_gravity="center_horizontal"
144	/>
145	
146	<TextView
147	android:id="@+id/song_lyrics"
148	android:layout_width="300dp"
149	
150	android:layout_height="wrap_content"
151	
152	app:layout_constraintStart_toStartOf="parent"
153	
154	app:layout_constraintEnd_toEndOf="parent"
155	android:layout_marginTop="8dp"
156	
157	app:layout_constraintTop_toBottomOf="@id/song_content"
158	
159	
160	android:layout_gravity="center_horizontal"
161	android:lineSpacingMultiplier="1.5"
162	android:text="llorem ipsum dolor
163	ist amenfdusbgjrabjfbshbfrabjrft
164	jfasjfksufheragjrgjjbgj
165	garhgiargjkafjkdgjdfjgdjkbgnajnngjd gaigirigoreio
166	igiarigreiogjraei garjgjewoiwetieri "
167	/>
168	</LinearLayout>

169	</androidx.core.widget.NestedScrollView>
170	
171	</androidx.constraintlayout.widget.ConstraintLayout
172	>

8. fragment_home.xml

Tabel 15. Source Code fragment_home.xml Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	xmlns:android="http://schemas.android.com/apk/res/an
4	droid"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	xmlns:app="http://schemas.android.com/apk/res-
8	auto"
9	xmlns:tools="http://schemas.android.com/tools"
10	tools:context=".HomeFragment">
11	
12	<androidx.appcompat.widget.Toolbar
13	android:id="@+id/toolbar"
14	android:layout_width="match_parent"
15	android:layout_height="wrap_content"
16	
17	android:background="@drawable/linear_gradient"
18	android:theme="?attr/actionBarTheme"
19	app:title="My Favourite Songs"
20	app:titleTextColor="@color/white"
21	app:layout_constraintTop_toTopOf="parent"
22	
23	app:layout_constraintStart_toStartOf="parent"

24	app:layout_constraintEnd_toEndOf="parent"
25	/>
26	
27	<androidx.recyclerview.widget.RecyclerView
28	android:id="@+id/rv_song"
29	android:layout_width="0dp"
30	android:layout_height="0dp"
31	android:layout_marginTop="85dp"
32	android:layout_marginLeft="20dp"
33	android:layout_marginRight="20dp"
34	android:layout_marginBottom="20dp"
35	
36	app:layout_constraintBottom_toBottomOf="parent"
37	app:layout_constraintEnd_toEndOf="parent"
38	
39	app:layout_constraintStart_toStartOf="parent"
40	app:layout_constraintTop_toTopOf="parent"
41	android:scrollbars="vertical"
42	/>
43	
44	</androidx.constraintlayout.widget.ConstraintLayout>

9. item_song.xml

Tabel 16. Source Code item_song.xml Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.cardview.widget.CardView
3	xmlns:android="http://schemas.android.com/apk/res/
4	android"
5	
6	

```

7
8 xmlns:card_view="http://schemas.android.com/apk/re
9 s-auto"
10     xmlns:tools="http://schemas.android.com/tools"
11     android:layout_width="match_parent"
12     android:layout_height="wrap_content"
13     android:id="@+id/card_view"
14     android:layout_gravity="center"
15     android:layout_marginStart="8dp"
16     android:layout_marginTop="4dp"
17     android:layout_marginEnd="8dp"
18     android:layout_marginBottom="8dp"
19     card_view:cardCornerRadius="20dp"
20     >
21
22
23 <androidx.constraintlayout.widget.ConstraintLayout
24     android:layout_width="match_parent"
25     android:layout_height="wrap_content"
26     android:padding="15dp">
27
28     <ImageView
29         android:id="@+id/img_cover"
30         android:layout_width="120dp"
31         android:layout_height="120dp"
32         android:scaleType="centerCrop"
33
34 card_view:layout_constraintBottom_toBottomOf="pare
35 nt"
36

```

```

37
38 card_view:layout_constraintStart_toStartOf="parent
39 "
40
41 card_view:layout_constraintTop_toTopOf="parent"
42         android:src="@drawable/place_holder"
43 />
44
45         <TextView
46             android:id="@+id/tv_title"
47             android:layout_width="180dp"
48             android:layout_height="wrap_content"
49             android:layout_marginTop="8dp"
50             android:layout_marginLeft="6dp"
51             android:textSize="16sp"
52             android:textStyle="bold"
53
54 card_view:layout_constraintEnd_toEndOf="parent"
55
56 card_view:layout_constraintHorizontal_bias="0.1"
57
58 card_view:layout_constraintStart_toEndOf="@id/img_
59 cover"
60
61 card_view:layout_constraintTop_toTopOf="parent"
62         tools:text="Jakarta Hari Ini"
63         android:padding="2dp"
64     />
65
66     <TextView

```

67	android:id="@+id/tv_desc"
68	android:layout_width="200dp"
69	android:layout_height="wrap_content"
70	android:textSize="12sp"
71	android:layout_marginTop="8dp"
72	android:layout_marginLeft="6.67dp"
73	android:padding="2dp"
74	
75	card_view:layout_constraintEnd_toEndOf="parent"
76	
77	card_view:layout_constraintHorizontal_bias="0.1"
78	
79	card_view:layout_constraintStart_toEndOf="@id/img_
80	cover"
81	
82	card_view:layout_constraintTop_toBottomOf="@id/tv_
83	title"
84	tools:text="a song by LANY, talk about
85	a person who lose his home, his comfort person....."
86	/>
87	
88	<androidx.appcompat.widget.AppCompatButton
89	android:id="@+id/button_detail"
90	android:layout_width="wrap_content"
91	android:layout_height="wrap_content"
92	android:layout_marginTop="16dp"
93	android:text="Detail"
94	android:textSize="12dp"
95	android:textColor="@color/white"
96	


```

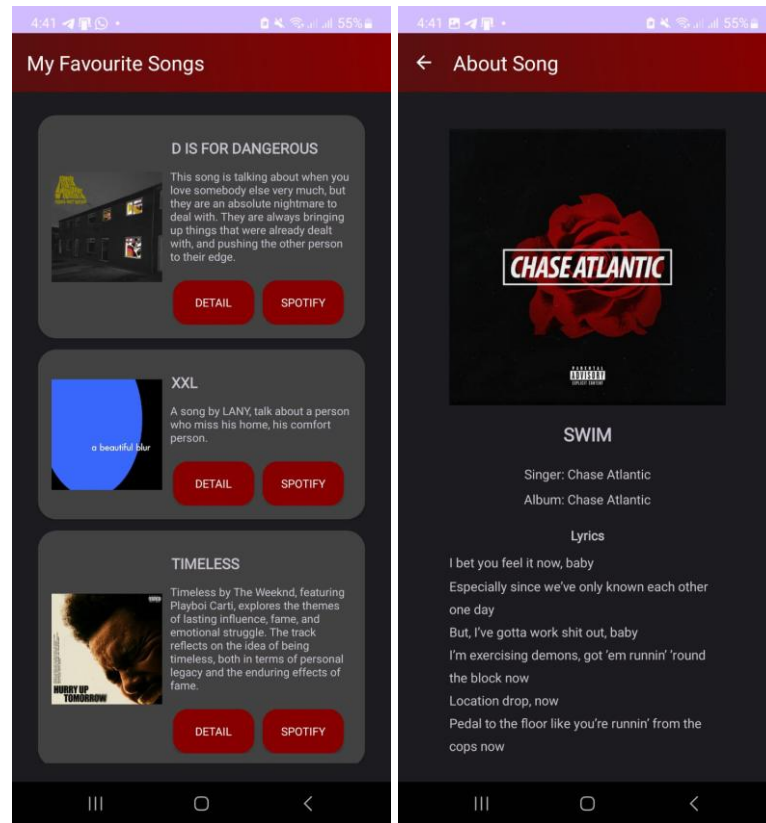
97
98 android:background="@drawable/button_background"
99
100 card_view:layout_constraintBottom_toBottomOf="pare
101 nt"
102
103 card_view:layout_constraintEnd_toEndOf="parent"
104
105 card_view:layout_constraintHorizontal_bias="0.1"
106
107 card_view:layout_constraintStart_toEndOf="@id/img_
108 cover"
109
110 card_view:layout_constraintTop_toBottomOf="@id/tv_
111 desc"
112
113 card_view:layout_constraintVertical_bias="0.0" />
114
115         <androidx.appcompat.widget.AppCompatButton
116             android:id="@+id/button_spotify"
117             android:layout_width="wrap_content"
118             android:layout_height="wrap_content"
119             android:layout_marginTop="16dp"
120             android:layout_marginLeft="100dp"
121             android:text="Spotify"
122             android:textSize="12dp"
123             android:textColor="@color/white"
124
125 android:background="@drawable/button_background"
126

```

```
127
128 card_view:layout_constraintBottom_toBottomOf="pare
129 nt"
130
131 card_view:layout_constraintEnd_toEndOf="parent"
132
133 card_view:layout_constraintStart_toEndOf="@id/img_
134 cover"
135
136 card_view:layout_constraintTop_toBottomOf="@id/tv_
137 desc"
138
139 card_view:layout_constraintVertical_bias="0.0" />
140
141 </androidx.constraintlayout.widget.ConstraintLayou
142 t>
143
144 </androidx.cardview.widget.CardView>
145
146
147
148
149
150
151
152
153
154
155
156
```

157	
158	
159	
160	
161	
162	
163	
164	
165	
166	
167	
168	
169	

B. Output Program



Gambar 7. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt

File ini merupakan logika utama dari aplikasi untuk menampilkan UI dan mengatur interaksi pengguna. Terdapat beberapa bagian penting, yaitu menambahkan HomeFragment sebagai tampilan awal saat aplikasi dijalankan. Terdapat beberapa bagian penting, di antaranya yaitu:

m) View Binding

```
private lateinit var binding: ActivityMainBinding
binding =
    ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
```

Digunakan untuk menghubungkan file layout activity_main.xml ke file MainActivity.kt tanpa perlu menggunakan findViewById. Sementara itu, binding.root akan menjadi tampilan utama yang ditampilkan di layar.

n) Fragment Manager

```
val fragmentManager = supportFragmentManager
```

Digunakan untuk mengelola fragment dalam activity. Lalu, supportFragmentManager merupakan salah satu bagian AndroidX yang mendukung fragment secara kompatibel di berbagai versi Android.

o) Memeriksa Fragment

```
val homeFragment = HomeFragment()
val fragment = fragmentManager.findFragmentByTag(HomeFragment::class.java.simpleName)
```

Digunakan untuk membuat instance HomeFragment. Kemudian, melakukan pemeriksaan apakah fragment dengan tag HomeFragment sudah ditambahkan sebelumnya, hal ini untuk mencegah fragment saat activity dipanggil ulang.

p) Menambahkan Fragment ke Activity

```
if (fragment !is HomeFragment) {
    Log.d("MyFlexibleFragment", "Fragment Name : "
+ HomeFragment::class.java.simpleName)
    fragmentManager
        .beginTransaction()
        .add(R.id.main, homeFragment,
HomeFragment::class.java.simpleName)
        .commit()
}
```

Jika fragment belum ada, maka akan ditambahkan ke dalam container R.id.main. Proses ini dilakukan menggunakan transaksi fragment.

`Log.d(...)` digunakan untuk mencatat nama fragment yang sedang ditambahkan ke logcat untuk keperluan debugging.

2. Song.kt

File ini mendefinisikan struktur data yang bernama Song. File ini digunakan untuk merepresentasikan informasi song/lagu secara lengkap dan dapat dikirim antar komponen Android. Terdapat beberapa bagian, yaitu:

g) Anotasi `@Parcelize`

```
@Parcelize
```

Anotasi ini digunakan untuk membuat objek Song dapat di-serialize secara otomatis menjadi bentuk yang dikirim melalui Intent atau Bundle. Digunakan bersama dengan interface `Parcelable`.

h) Deklarasi

```
data class Song(  
    val title: String,  
    val link: String,  
    val photo: String,  
    val singer: String,  
    val album: String,  
    val lyrics: String,  
    val desc: String  
) : Parcelable
```

Bagian ini mendeklarasikan data class.

i) `Parcelable`

```
) : Parcelable
```

Bagian ini membuat Song dapat dikirim melalui Intent dan Bundle.

3. activity_main.xml

File ini merupakan Fragment yang digunakan untuk menampilkan detail dari sebuah lagu yang dipilih. Fragment ini menerima data dari argument Bundle

dan menampilkannya di layout `fragment_detail.xml`. Terdapat beberapa bagian, yaitu:

i) View Binding pada Fragment

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

Digunakan untuk mencegah memory leak, di mana `_binding` hanya aktif selama `onCreateView` hingga `onDestroyView` dan `binding.get()` digunakan untuk mengakses binding secara aman selama fragment aktif.

j) `onCreateView()`

```
_binding = FragmentDetailBinding.inflate(inflater,
    container, false)
```

Digunakan untuk meng-inflate layout fragment menggunakan view binding. `Lau, binding.root` akan dikembalikan sebagai tampilan utama fragment.

k) Menerima Argument dari Fragment Lain

```
val title = arguments?.getString("TITLE")
val photo = arguments?.getString("PHOTO")
val singer = arguments?.getString("SINGER")
val album = arguments?.getString("ALBUM")
val lyrics = arguments?.getString("LYRICS")
```

Digunakan untuk menerima data berupa `String` dari Fragment sebelumnya menggunakan `Bundle`.

l) Menampilkan Data ke UI

```
binding.songTitle.text = title
binding.songSinger.text = singer
binding.songAlbum.text = album
binding.songLyrics.text = lyrics
```

Menampilkan data song ke dalam `TextView` di layout.

m) Menampilkan Gambar dengan Glide

```
photo?.let {  
    Glide.with(requireContext())  
        .load(it)  
        .into(binding.songCover)  
}
```

Digunakan untuk menampilkan gambar dari URL ke dalam Image View songCover menggunakan library Glide.

n) Tombol Back di Toolbar

```
binding.toolbar.setNavigationOnClickListener {  
    parentFragmentManager.popBackStack()  
}
```

Digunakan untuk mengatur aksi tombol back di toolbar agar Kembali ke Fragment sebelumnya.

o) onDestroyView()

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

Digunakan untuk membersihkan objek binding saat fragment dihancurkan agar tidak terjadi memory leak.

4. HomeFragment.kt

File ini merupakan tampilan utama dari aplikasi yang menampilkan daftar song dalam bentuk RecyclerView. Terdapat beberapa bagian, yaitu:

a) View Binding pada Fragment

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

Digunakan untuk mengakses elemen XML (fragment_home.xml) secara langsung dan aman dari NullPointerException.

b) Deklarasi Variabel

```
private lateinit var songAdapter: ListSongAdapter
private val list = ArrayList<Song>()
```

Di sini terdapat adapter khusus untuk menampilkan data song dalam RecyclerView dan list digunakan untuk menyimpan data lagu dari resource (array).

c) onCreateView()

```
list.clear()
list.addAll(getListSong())
setupRecyclerView()
```

Digunakan untuk mengambil data dari array dan menyimpannya dala list. Kemudian, memanggil fungsi setupRecyclerView() untuk menampilkan daftar lagu ke layar.

d) setupRecyclerView()

```
songAdapter = ListSongAdapter(
    list,
    onSpotifyClick = { link -> ... },
    onDetailClick = { title, photo, ... -> ... }
)
```

onSpotifyClick: ketika tombol Spotify di item lagu ditekan, membuka link ke aplikasi/browser Spotify.

onDetailClick: ketika item lagu ditekan, pindah ke DetailFragment dan mengirim data lagu lewat arguments.

e) getListSong()

```
val dataTitle =
resources.getStringArray(R.array.data_title)
...
```

Digunakan untuk mengambil semua data lagu dari file strings.xml (berupa array resource). Kemudian, data dikonversi menjadi objek Song dan dimasukkan ke list.

f) Navigasi ke DetailFragment

```
val detailFragment = DetailFragment().apply {  
    arguments = Bundle().apply {  
        putString("TITLE", title)  
        ...  
    }  
}
```

Digunakan untuk membuat instance detailFragment, lalu mengirim data melalui arguments. Kemudian, tampilan diubah dengan DetailFragment menggunakan FragmentTransaction.

g) onDestroyView()

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

Digunakan untuk membersihkan binding saat tampilan dihancurkan untuk mencegah memory leak

5. ListSongAdapter.kt

File ini adalah adapter untuk RecyclerView yang bertanggung jawab menampilkan daftar lagu dalam tampilan list. Adapter ini juga menangani event klik tombol Spotify dan tombol Detail untuk setiap item lagu. Terdapat beberapa bagian, yaitu:

a) Deklarasi Class dan Konstruktor

```
class ListSongAdapter(  
    private val listSong: ArrayList<Song>,  
    private val onSpotifyClick: (String) -> Unit,  
    private val onDetailClick: (String, String,  
    String, String, String) -> Unit)
```

```

:
RecyclerView.Adapter<ListSongAdapter.ListViewHold
er>()
listSong merupakan dat berisi objek Song.
onSpotifyClick merupakan function yang dipanggil saat tombol
Spotify diklik.
onDetailClick merupakan function saat tombol Detail diklik dan data
lagu dikirim ke fragment detail.

```

b) ViewHolder

```

inner class ViewHolder(val binding:
ItemSongBinding) :
RecyclerView.ViewHolder(binding.root)
ViewHolder menyimpan referensi ke layout item lagu (item_song.xml)
menggunakan View Binding.

```

c) bind()

```

fun bind(song: Song) {
    binding.tvTitle.text = song.title
    binding.tvDesc.text = song.desc
    Glide.with(binding.root.context)
        .load(song.photo)
        .into(binding.imgCover)

    binding.buttonSpotify.setOnClickListener {
        onSpotifyClick(song.link)
    }

    binding.buttonDetail.setOnClickListener {
        onDetailClick(song.title, song.photo,
song.singer, song.album, song.lyrics)
    }
}

```

```
}
```

tvTitle dan tvDesc digunakan untuk menampilkan judul dan deskripsi song.

Glide digunakan untuk memuat dan menampilkan gambar cover song dari URL.

buttonSpotify digunakan untuk membuka link Spotify.

buttonDetail digunakan untuk mengirim data lagu ke DetailFragment.

d) onCreateViewHolder()

```
val binding =  
ItemSongBinding.inflate(LayoutInflater.from(paren  
t.context), parent, false)  
return ViewHolder(binding)
```

Bagian ini berfungsi membuat tampilan satu item daftar lagu menggunakan item_song.xml dan mengembalikan ViewHolder.

e) getItemCount()

```
override fun getItemCount(): Int = listSong.size
```

Bagian ini berfungsi untuk menentukan jumlah item dalam list, sesuai dengan jumlah data song.

f) onBindViewHolder

```
holder.bind(listSong[position])
```

Digunakan untuk memanggil fungsi bind() untuk menampilkan data song pada posisi yang sesuai di RecyclerView.

6. activity_main.xml

activity_main.xml adalah tata letak kosong dengan FrameLayout, dirancang agar fleksibel menampilkan konten fragment. Ini umum digunakan dalam aplikasi Android berbasis fragment agar UI dapat berganti tanpa membuat banyak activity.

7. Fragment_detail.xml

Terdapat beberapa bagian penting, yaitu:

a) Root Layout

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

ConstraintLayout digunakan sebagai root agar kita bisa menempatkan elemen UI secara fleksibel dengan constraint.

b) Toolbar

```
<androidx.appcompat.widget.Toolbar ... />
```

Digunakan untuk membuat custom toolbar.

c) NestedScrollView

```
<androidx.core.widget.NestedScrollView ... >
```

Digunakan untuk membungkus konten agar bisa di-scroll secara vertikal.

d) LinearLayout

Di dalam NestedScrollView, digunakan LinearLayout vertikal untuk menampilkan isi konten.

8. Fragment_home.kt

a) Root

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

Memungkinkan penempatan elemen UI secara fleksibel dengan constraint antar elemen.

b) Toolbar

```
<androidx.appcompat.widget.Toolbar ... />
```

Bagian ini dapat menampilkan judul halaman, yaitu “My Favourite Song”.

c) RecyclerView

```
<androidx.recyclerview.widget.RecyclerView ... />
```

Menampilkan list song secara vertikal (akan diatur melalui adapter di kode Kotlin).

9. item_song.xml

a) Root: CardView

```
<androidx.cardview.widget.CardView ...>
```

Merupakan pembungkus utama yang membentuk tampilan item seperti kartu dengan sudut melengkung (`cardCornerRadius="20dp"`).

b) ConstraintLayout

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

Digunakan untuk mengatur komponen di dalam CardView secara fleksibel.

SOAL 2

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

A. Pembahasan

RecyclerView masih banyak digunakan karena memiliki API yang matang. Tidak semua pengembang aplikasi dapat menggunakan dan mau beralih Jetpack Compose walaupun lebih mudah, hal ini karena mereka sudah terbiasa. Kemudian, jika pengembang ingin membangun sebuah aplikasi yang mendukung perangkat minSDK di bawah 21, maka RecyclerView masih sangat relevan (Compose membutuhkan API 21 ke atas).

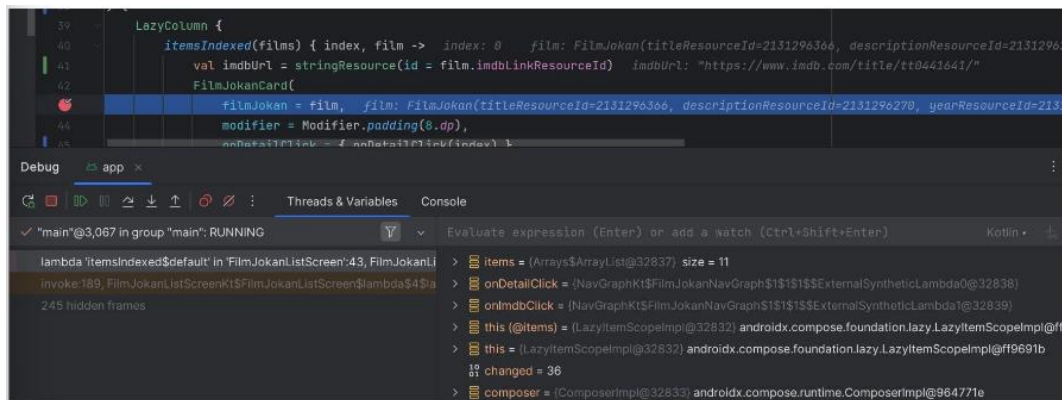
MODUL 4: VIEWMODEL & DEBUGGING

SOAL 1

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:
 - a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
 - b. Gunakan ViewModelFactory dalam pembuatan ViewModel.
 - c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment.
 - d. Gunakan logging untuk event berikut.
 - a. Log saat data item masuk ke dalam list
 - b. Log saat tombol Detail dan tombol Explicit Intent ditekan
 - c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
 - e. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi.

Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step, Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out
2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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



A. Source Code

1. MainActivity.kt

Tabel 17. Source Code MainActivity.kt Soal 1

1	package com.example.scrollablelist
2	
3	import android.os.Bundle
4	import android.util.Log
5	import androidx.activity.enableEdgeToEdge
6	import androidx.appcompat.app.AppCompatActivity
7	import androidx.core.view.ViewCompat
8	import androidx.core.view.WindowInsetsCompat
9	import
10	com.example.scrollablelist.databinding.ActivityMainB
11	inding
12	
13	class MainActivity : AppCompatActivity() {
14	private lateinit var binding:
15	ActivityMainBinding
16	
17	override fun onCreate(savedInstanceState:
18	Bundle?) {
19	super.onCreate(savedInstanceState)

20	
21	binding =
22	ActivityMainBinding.inflate(layoutInflater)
23	setContentView(binding.root)
24	
25	val fragmentManager = supportFragmentManager
26	val homeFragment = HomeFragment()
27	val fragment =
28	fragmentManager.findFragmentByTag(HomeFragment::class.java.simpleName)
29	
30	if (fragment !is HomeFragment) {
31	Log.d("MyFlexibleFragment", "Fragment
32	Name : " + HomeFragment::class.java.simpleName)
33	fragmentManager
34	.beginTransaction()
35	.add(R.id.main, homeFragment,
36	HomeFragment::class.java.simpleName)
37	.commit()
38	}
39	}
40	}

2. Song.kt

Tabel 18. Source Code Song.kt Soal 1

1	package com.example.scrollablelist
2	
3	import android.icu.text.CaseMap.Title
4	import android.os.Parcelable
5	import kotlinx.parcelize.Parcelize
6	

7	@Parcelize
8	data class Song(
9	val title: String,
10	val link: String,
11	val photo: String,
12	val singer: String,
13	val album: String,
14	val lyrics: String,
15	val desc: String
16	
17):Parcelable

3. DetailFragment.kt

Tabel 19. Source Code DetailFragment.kt Soal 1

1	package com.example.scrollablelist
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 com.bumptech.glide.Glide
9	import
10	com.example.scrollablelist.databinding.FragmentDetailBinding
11	
12	class DetailFragment : Fragment() {
13	
14	private var _binding: FragmentDetailBinding? = null
15	private val binding get() = _binding!!
16	

```

17         override fun onCreateView(
18             inflater: LayoutInflater,
19             container: ViewGroup?,
20             savedInstanceState: Bundle?
21         ): View? {
22             _binding = FragmentDetailBinding.inflate(inflater,
23 container, false)
24
25             val title = arguments?.getString("TITLE")
26             val photo = arguments?.getString("PHOTO")
27             val singer = arguments?.getString("SINGER")
28             val album = arguments?.getString("ALBUM")
29             val lyrics = arguments?.getString("LYRICS")
30
31             binding.songTitle.text = title
32             photo?.let {
33                 Glide.with(requireContext())
34                     .load(it)
35                     .into(binding.songCover)
36             }
37             binding.songSinger.text = singer
38             binding.songAlbum.text = album
39             binding.songLyrics.text = lyrics
40
41             binding.toolbar.setNavigationOnClickListener {
42                 parentFragmentManager.popBackStack()
43             }
44
45             return binding.root
46         }

```

47	
48	override fun onDestroyView() {
49	super.onDestroyView()
50	_binding = null
51	}
52	}

4. HomeFragment.kt

Tabel 20. Source Code HomeFragment.kt Soal 1

1	package com.example.scrollablelist.home
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
13	androidx.recyclerview.widget.LinearLayoutManager
14	import com.example.scrollablelist.R
15	import
16	com.example.scrollablelist.adapter.ListSongAdapter
17	import
18	com.example.scrollablelist.databinding.FragmentHomeB
19	inding
20	import
21	com.example.scrollablelist.detail.DetailFragment

```

22 import com.example.scrollablelist.model.Song
23 import
24 com.example.scrollablelist.utils.HomeViewModelFactor
25 y
26 import kotlinx.coroutines.flow.collectLatest
27 import kotlinx.coroutines.launch
28 import android.util.Log
29
30 class HomeFragment : Fragment() {
31     private var _binding: FragmentHomeBinding? =
32     null
33     private val binding get() = _binding!!
34
35     private lateinit var songAdapter:
36 ListSongAdapter
37
38     private val viewModel: HomeViewModel by
39 viewModel {
40         HomeViewModelFactory(resources)
41     }
42
43     override fun onCreateView(
44         inflater: LayoutInflater,
45         container: ViewGroup?,
46         savedInstanceState: Bundle?
47     ): View? {
48         _binding =
49 FragmentHomeBinding.inflate(inflater, container,
50 false)
51

```

```

52         return binding.root
53     }
54
55     override fun onCreateView(view: View,
56 savedInstanceState: Bundle?) {
57         super.onCreateView(view,
58 savedInstanceState)
59
60         setupRecyclerView()
61         observeSongList()
62
63         viewModel.loadSongs()
64     }
65
66     private fun setupRecyclerView() {
67         songAdapter = ListSongAdapter(
68             ArrayList(),
69             onSpotifyClick = { link ->
70                 Log.d("HomeFragment", "Spotify
71 button clicked for link: $link")
72                 Log.e("Intent to Spotify", "Going to
73 $link")
74                 val intent =
75 Intent(Intent.ACTION_VIEW, Uri.parse(link))
76                 startActivity(intent)
77             },
78             onDetailClick = { title, photo, singer,
79 album, lyrics ->
80                 Log.d(
81                     "HomeFragment",

```

82	"Detail button clicked - Title:
83	\$title, Singer: \$singer, Album: \$album"
84)
85	val detailFragment =
86	DetailFragment().apply {
87	arguments = Bundle().apply {
88	putString("TITLE", title)
89	putString("PHOTO", photo)
90	putString("SINGER", singer)
91	putString("ALBUM", album)
92	putString("LYRICS", lyrics)
93	}
94	}
95	
96	
97	parentFragmentManager.beginTransaction()
98	.replace(R.id.main,
99	detailFragment)
100	.addToBackStack(null)
101	.commit()
102	}
103)
104	
105	binding.rvSong.apply {
106	layoutManager =
107	LinearLayoutManager(context)
108	adapter = songAdapter
109	setHasFixedSize(true)
110	}
111	}

112	
113	private fun observeSongList() {
114	viewLifecycleOwner.lifecycleScope.launch {
	viewModel.songList.collectLatest { list
	->
	songAdapter.setData(list)
	}
	}
	override fun onDestroyView() {
	super.onDestroyView()
	_binding = null
	}
	}

5. HomeViewModel.kt

Tabel 21. Source Code HomeViewModel.kt Soal 1

1	package com.example.scrollablelist.home
2	
3	import android.content.res.Resources
4	import androidx.lifecycle.ViewModel
5	import androidx.lifecycle.viewModelScope
6	import com.example.scrollablelist.R
7	import com.example.scrollablelist.model.Song
8	import kotlinx.coroutines.flow.Flow
9	import kotlinx.coroutines.flow.MutableStateFlow
10	import kotlinx.coroutines.flow.StateFlow
11	import kotlinx.coroutines.flow.flow
12	import kotlinx.coroutines.flow.onStart

```

13 import kotlinx.coroutines.launch
14 import android.util.Log
15
16 class HomeViewModel(private val resources:
17 Resources) : ViewModel() {
18     //MutableStateFlow untuk menyimpan list
19     item(private)
20     private val _songList =
21     MutableStateFlow<List<Song>>(emptyList())
22
23     //public read-only access untuk mengobservasi
24     list item
25     val songList: StateFlow<List<Song>> get() =
26     _songList
27
28     //function yang mengembalikan data list chara
29     private fun getSongFlow(): Flow<List<Song>> =
30     flow {
31         //mengambil data dari Resources
32         val dataTitle =
33         resources.getStringArray(R.array.data_title)
34         val dataLink =
35         resources.getStringArray(R.array.data_link)
36         val dataPhoto =
37         resources.getStringArray(R.array.data_photo)
38         val dataSinger =
39         resources.getStringArray(R.array.data_singer)
40         val dataAlbum =
41         resources.getStringArray(R.array.data_album)
42

```

```

43         val dataLyrics =
44         resources.getStringArray(R.array.data_lyrics)
45         val dataDesc =
46         resources.getStringArray(R.array.data_desc)
47
48         //membuat list
49         val listSong = ArrayList<Song>()
50         for (i in dataTitle.indices) {
51             val song = Song(
52                 title = dataTitle[i],
53                 link = dataLink[i],
54                 photo = dataPhoto[i],
55                 singer = dataSinger[i],
56                 album = dataAlbum[i],
57                 lyrics = dataLyrics[i],
58                 desc = dataDesc[i]
59             )
60             listSong.add(song)
61         }
62         //mengirim item ke collector
63         emit(listSong)
64     }
65     //function load item
66     fun loadSongs() {
67         viewModelScope.launch {
68             collectSongs()
69         }
70     }
71
72     private suspend fun collectSongs() {

```

73	getSongFlow()
74	.onStart {
75	_songList.value = emptyList()
76	}
77	.collect { songs ->
78	Log.d("HomeViewModel", "Songs
79	loaded: \${songs.size} items")
80	_songList.value = songs
81	}
82	}
83	
84	}

6. ListSongAdapter.kt

Tabel 22. Source Code ListSongAdapter.kt Soal 1

1	package com.example.scrollablelist
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
8	com.example.scrollablelist.databinding.ItemSongBinding
9	
10	class ListSongAdapter(
11	private val listSong: ArrayList<Song>,
12	private val onSpotifyClick: (String) -> Unit,
13	private val onDetailClick: (String, String, String,
14	String, String) -> Unit)
15	

```

16         : RecyclerView.Adapter<ListSongAdapter.ListViewHolder>()
17     {
18
19         inner class ListViewHolder(val binding: ItemSongBinding)
20     : RecyclerView.ViewHolder(binding.root) {
21         fun bind(song: Song) {
22             binding.tvTitle.text = song.title
23             binding.tvDesc.text = song.desc
24             Glide.with(binding.root.context)
25                 .load(song.photo)
26                 .into(binding.imgCover)
27
28
29             binding.buttonSpotify.setOnClickListener {
30                 onSpotifyClick(song.link)
31             }
32
33             binding.buttonDetail.setOnClickListener {
34                 onDetailClick(song.title, song.photo,
35 song.singer, song.album, song.lyrics)
36             }
37         }
38     }
39
40     override fun onCreateViewHolder(parent: ViewGroup,
41 viewType: Int): ListViewHolder {
42         val binding =
43 ItemSongBinding.inflate(LayoutInflater.from(parent.context),
44 parent, false)
45         return ListViewHolder(binding)

```

46	}
47	
48	override fun getItemCount(): Int {
49	return listSong.size
50	}
51	
52	override fun onBindViewHolder(holder: ListViewHolder,
53	position: Int) {
54	holder.bind(listSong[position])
55	}
	}

7. activity_main.xml

Tabel 23. Source Code activity_main.xml Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<FrameLayout
3	xmlns:android="http://schemas.android.com/apk/res/andr
4	oid"
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	tools:context=".MainActivity">
1	</FrameLayout>
0	

8. fragment_detail.xml

Tabel 24. Source Code fragment_detail.xml Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	

3	<androidx.constraintlayout.widget.ConstraintLayout
4	xmlns:android="http://schemas.android.com/apk/res/android"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	xmlns:app="http://schemas.android.com/apk/res-auto"
8	xmlns:tools="http://schemas.android.com/tools"
9	tools:context=".DetailFragment">
10	
11	<androidx.appcompat.widget.Toolbar
12	android:id="@+id/toolbar"
13	android:layout_width="match_parent"
14	android:layout_height="wrap_content"
15	android:background="@drawable/linear_gradient"
16	android:theme="?attr/actionBarTheme"
17	app:title="About Song"
18	app:titleTextColor="@color/white"
19	app:navigationIcon="@drawable/arrow_back"
20	app:layout_constraintTop_toTopOf="parent"
21	app:layout_constraintStart_toStartOf="parent"
22	app:layout_constraintEnd_toEndOf="parent"
23	/>
24	
25	<androidx.core.widget.NestedScrollView
26	android:id="@+id/scroll_view"
27	android:layout_width="0dp"
28	android:layout_height="0dp"
29	app:layout_constraintTop_toBottomOf="@id/toolbar"
30	app:layout_constraintBottom_toBottomOf="parent"
31	app:layout_constraintStart_toStartOf="parent"
32	app:layout_constraintEnd_toEndOf="parent"

33	android:scrollbars="vertical"
34	android:padding="16dp"
35	android:layout_margin="8dp"
36	>
37	
38	<LinearLayout
39	android:layout_width="match_parent"
40	android:layout_height="wrap_content"
41	android:orientation="vertical"
42	android:padding="8dp">
43	
44	<ImageView
45	android:id="@+id/song_cover"
46	android:layout_width="300dp"
47	android:layout_height="300dp"
48	android:layout_marginTop="8dp"
49	android:src="@drawable/place_holder"
50	android:layout_gravity="center_horizontal"
51	/>
52	
53	<TextView
54	android:id="@+id/song_title"
55	android:layout_width="wrap_content"
56	android:layout_height="wrap_content"
57	
58	app:layout_constraintStart_toStartOf="parent"
59	app:layout_constraintEnd_toEndOf="parent"
60	
61	app:layout_constraintTop_toBottomOf="@id/song_cover"
62	android:layout_marginTop="16dp"

63	android:text="TEDDY PICKER"
64	android:textStyle="bold"
65	android:textSize="20sp"
66	android:padding="2dp"
67	android:layout_gravity="center_horizontal"
68	/>
69	
70	<TextView
71	android:id="@+id/song_singer"
72	android:layout_width="wrap_content"
73	android:layout_height="wrap_content"
74	
75	app:layout_constraintStart_toStartOf="parent"
76	app:layout_constraintEnd_toEndOf="parent"
77	
78	app:layout_constraintTop_toBottomOf="@id/song_title"
79	android:layout_marginTop="16dp"
80	android:padding="2dp"
81	android:text="Singer: Arctic Monkeys"
82	android:textSize="14sp"
83	android:layout_gravity="center_horizontal"
84	/>
85	
86	<TextView
87	android:id="@+id/song_album"
88	android:layout_width="wrap_content"
89	android:layout_height="wrap_content"
90	
91	app:layout_constraintStart_toStartOf="parent"
92	app:layout_constraintEnd_toEndOf="parent"

93	android:layout_marginTop="4dp"
94	android:padding="2dp"
95	
96	app:layout_constraintTop_toBottomOf="@id/song_singer"
97	android:text="Album: Favourite Worst
98	Nightmare"
99	android:textSize="14sp"
100	android:layout_gravity="center_horizontal"
101	/>
102	
103	<TextView
104	android:id="@+id/song_content"
105	android:layout_width="wrap_content"
106	android:layout_height="wrap_content"
107	
108	app:layout_constraintStart_toStartOf="parent"
109	app:layout_constraintEnd_toEndOf="parent"
110	android:layout_marginTop="16dp"
111	android:padding="2dp"
112	
113	app:layout_constraintTop_toBottomOf="@id/song_album"
114	android:text="Lyrics"
115	android:textStyle="bold"
116	android:textSize="14sp"
117	android:layout_gravity="center_horizontal"
118	/>
119	
120	<TextView
121	android:id="@+id/song_lyrics"
122	android:layout_width="300dp"

123	android:layout_height="wrap_content"
124	
125	app:layout_constraintStart_toStartOf="parent"
126	app:layout_constraintEnd_toEndOf="parent"
127	android:layout_marginTop="8dp"
128	
129	app:layout_constraintTop_toBottomOf="@id/song_content"
130	android:layout_gravity="center_horizontal"
131	android:lineSpacingMultiplier="1.5"
132	android:text="llorem ipsum dolor ist
133	amenfdusbgjrabjfbshbfrabjrft jfasjfkufheragjrgjjbgj
134	garhgiargjkafjkdgdjfdjgdkbggnajngjd gaigirigoreio
135	igiarigreiogjraei garjggewoiwetieri "
136	/>
137	</LinearLayout>
138	</androidx.core.widget.NestedScrollView>
139	
140	</androidx.constraintlayout.widget.ConstraintLayout>

9. fragment_home.xml

Tabel 24. Source Code fragment_home.xml Soal 1

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

```

10 <androidx.appcompat.widget.Toolbar
11     android:id="@+id/toolbar"
12     android:layout_width="match_parent"
13     android:layout_height="wrap_content"
14     android:background="@drawable/linear_gradient"
15     android:theme="?attr/actionBarTheme"
16     app:title="My Favourite Songs"
17     app:titleTextColor="@color/white"
18     app:layout_constraintTop_toTopOf="parent"
19     app:layout_constraintStart_toStartOf="parent"
20     app:layout_constraintEnd_toEndOf="parent"
21 />
22
23 <androidx.recyclerview.widget.RecyclerView
24     android:id="@+id/rv_song"
25     android:layout_width="0dp"
26     android:layout_height="0dp"
27     android:layout_marginTop="85dp"
28     android:layout_marginLeft="20dp"
29     android:layout_marginRight="20dp"
30     android:layout_marginBottom="20dp"
31     app:layout_constraintBottom_toBottomOf="parent"
32     app:layout_constraintEnd_toEndOf="parent"
33     app:layout_constraintStart_toStartOf="parent"
34     app:layout_constraintTop_toTopOf="parent"
35     android:scrollbars="vertical"
36 />
37
38 </androidx.constraintlayout.widget.ConstraintLayout>

```

10. item_song.xml

Tabel 25. Source Code item_song.xml Soal 1

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.cardview.widget.CardView
3	xmlns:android="http://schemas.android.com/apk/res/an
4	droid"
5	
6	xmlns:card_view="http://schemas.android.com/apk/res-
7	auto"
8	xmlns:tools="http://schemas.android.com/tools"
9	android:layout_width="match_parent"
10	android:layout_height="wrap_content"
11	android:id="@+id/card_view"
12	android:layout_gravity="center"
13	android:layout_marginStart="8dp"
14	android:layout_marginTop="4dp"
15	android:layout_marginEnd="8dp"
16	android:layout_marginBottom="8dp"
17	card_view:cardCornerRadius="20dp"
18	>
19	
20	
21	<androidx.constraintlayout.widget.ConstraintLayout
22	android:layout_width="match_parent"
23	android:layout_height="wrap_content"
24	android:padding="15dp">
25	
26	<ImageView
27	android:id="@+id/img_cover"
28	android:layout_width="120dp"

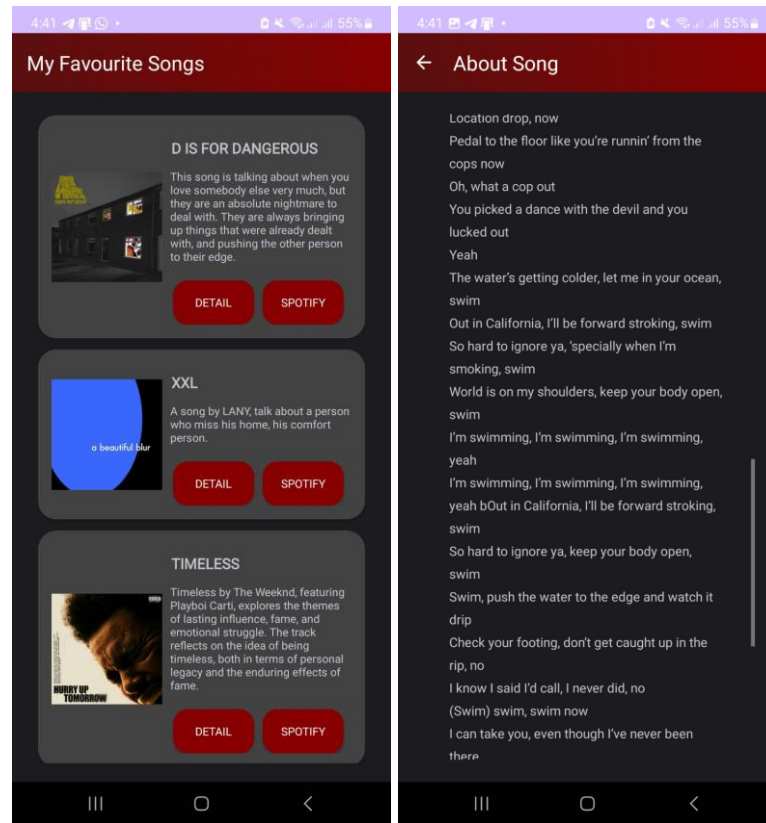
29	android:layout_height="120dp"
30	android:scaleType="centerCrop"
31	
32	card_view:layout_constraintBottom_toBottomOf="parent"
33	"
34	
35	card_view:layout_constraintStart_toStartOf="parent"
36	
37	card_view:layout_constraintTop_toTopOf="parent"
38	android:src="@drawable/place_holder" />
39	
40	<TextView
41	android:id="@+id/tv_title"
42	android:layout_width="180dp"
43	android:layout_height="wrap_content"
44	android:layout_marginTop="8dp"
45	android:layout_marginLeft="6dp"
46	android:textSize="16sp"
47	android:textStyle="bold"
48	
49	card_view:layout_constraintEnd_toEndOf="parent"
50	
51	card_view:layout_constraintHorizontal_bias="0.1"
52	
53	card_view:layout_constraintStart_toEndOf="@id/img_co
54	ver"
55	
56	card_view:layout_constraintTop_toTopOf="parent"
57	tools:text="Jakarta Hari Ini"
58	android:padding="2dp"

59	<code></></code>
60	
61	<code><TextView</code>
62	<code> android:id="@+id/tv_desc"</code>
63	<code> android:layout_width="200dp"</code>
64	<code> android:layout_height="wrap_content"</code>
65	<code> android:textSize="12sp"</code>
66	<code> android:layout_marginTop="8dp"</code>
67	<code> android:layout_marginLeft="6.67dp"</code>
68	<code> android:padding="2dp"</code>
69	
70	<code>card_view:layout_constraintEnd_toEndOf="parent"</code>
71	
72	<code>card_view:layout_constraintHorizontal_bias="0.1"</code>
73	
74	<code>card_view:layout_constraintStart_toEndOf="@id/img_co</code>
75	<code>ver"</code>
76	
77	<code>card_view:layout_constraintTop_toBottomOf="@id/tv_ti</code>
78	<code>tle"</code>
79	<code> tools:text="a song by LANY, talk about a</code>
80	<code>person who lose his home, his comfort person....."</code>
81	<code></></code>
82	
83	<code><androidx.appcompat.widget.AppCompatButton</code>
84	<code> android:id="@+id/button_detail"</code>
85	<code> android:layout_width="wrap_content"</code>
86	<code> android:layout_height="wrap_content"</code>
87	<code> android:layout_marginTop="16dp"</code>
88	<code> android:text="Detail"</code>

89	android:textSize="12dp"
90	android:textColor="@color/white"
91	
92	android:background="@drawable/button_background"
93	
94	card_view:layout_constraintBottom_toBottomOf="parent"
95	"
96	
97	card_view:layout_constraintEnd_toEndOf="parent"
98	
99	card_view:layout_constraintHorizontal_bias="0.1"
100	
101	card_view:layout_constraintStart_toEndOf="@id/img_co
102	ver"
103	
104	card_view:layout_constraintTop_toBottomOf="@id/tv_de
105	sc"
106	
107	card_view:layout_constraintVertical_bias="0.0" />
108	
109	<androidx.appcompat.widget.AppCompatButton
110	android:id="@+id/button_spotify"
111	android:layout_width="wrap_content"
112	android:layout_height="wrap_content"
113	android:layout_marginTop="16dp"
114	android:layout_marginLeft="100dp"
115	android:text="Spotify"
116	android:textSize="12dp"
117	android:textColor="@color/white"
118	

119	
120	android:background="@drawable/button_background"
121	
122	card_view:layout_constraintBottom_toBottomOf="parent
123	"
124	
125	card_view:layout_constraintEnd_toEndOf="parent"
126	
127	card_view:layout_constraintStart_toEndOf="@id/img_co
128	ver"
129	
130	card_view:layout_constraintTop_toBottomOf="@id/tv_de
131	sc"
132	
133	card_view:layout_constraintVertical_bias="0.0" />
134	
135	</androidx.constraintlayout.widget.ConstraintLayout>
136	
137	</androidx.cardview.widget.CardView>

B. Output Program



Gambar 8. Screenshot Hasil Jawaban Soal 1



Gambar 9. Screenshot Timber Button Soal 1 (2)

C. Pembahasan

1. MainActivity.kt

File ini merupakan logika utama dari aplikasi untuk menampilkan UI dan mengatur interaksi pengguna. Terdapat beberapa bagian penting, yaitu menambahkan HomeFragment sebagai tampilan awal saat aplikasi dijalankan. Terdapat beberapa bagian penting, di antaranya yaitu:

a) View Binding

```
private lateinit var binding: ActivityMainBinding
binding =
    ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
```

Digunakan untuk menghubungkan file layout activity_main.xml ke file MainActivity.kt tanpa perlu menggunakan findViewById. Sementara itu, binding.root akan menjadi tampilan utama yang ditampilkan di layar.

b) Fragment Manager

```
val fragmentManager = supportFragmentManager
```

Digunakan untuk mengelola fragment dalam activity. Lalu, supportFragmentManager merupakan salah satu bagian AndroidX yang mendukung fragment secara kompatibel di berbagai versi Android.

c) Memeriksa Fragment

```
val homeFragment = HomeFragment()
val fragment = fragmentManager.findFragmentByTag(HomeFragment::class.java.simpleName)
```

Digunakan untuk membuat instance HomeFragment. Kemudian, melakukan pemeriksaan apakah fragment dengan tag HomeFragment sudah ditambahkan sebelumnya, hal ini untuk mencegah fragment saat activity dipanggil ulang.

d) Menambahkan Fragment ke Activity

```
if (fragment !is HomeFragment) {
    Log.d("MyFlexibleFragment", "Fragment Name : "
        + HomeFragment::class.java.simpleName)
    fragmentManager
        .beginTransaction()
```

```

        .add(R.id.main, homeFragment,
HomeFragment::class.java.simpleName)
        .commit()
    }

```

Jika fragment belum ada, maka akan ditambahkan ke dalam container R.id.main. Proses ini dilakukan menggunakan transaksi fragment. Log.d(...) digunakan untuk mencatat nama fragment yang sedang ditambahkan ke logcat untuk keperluan debugging.

2. Song.kt

File ini mendefinisikan struktur data yang bernama Song. File ini digunakan untuk merepresentasikan informasi song/lagu secara lengkap dan dapat dikirim antar komponen Android. Terdapat beberapa bagian, yaitu:

a) Anotasi @Parcelize

```
@Parcelize
```

Anotasi ini digunakan untuk membuat objek Song dapat di-serialize secara otomatis menjadi bentuk yang dikirim melalui Intent atau Bundle. Digunakan bersama dengan interface Parcelable.

b) Deklarasi

```

data class Song(
    val title: String,
    val link: String,
    val photo: String,
    val singer: String,
    val album: String,
    val lyrics: String,
    val desc: String
) : Parcelable

```

Bagian ini mendeklarasikan data class.

c) Parcelable

```
) : Parcelable
```

Bagian ini membuat Song dapat dikirim melalui Intent dan Bundle.

3. activity_main.xml

File ini merupakan Fragment yang digunakan untuk menampilkan detail dari sebuah lagu yang dipilih. Fragment ini menerima data dari argument Bundle dan menampilkannya di layout fragment_detail.xml. Terdapat beberapa bagian, yaitu:

a) View Binding pada Fragment

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

Digunakan untuk mencegah memory leak, di mana `_binding` hanya aktif selama `onCreateView` hingga `onDestroyView` dan `binding.get()` digunakan untuk mengakses binding secara aman selama fragment aktif.

b) onCreateView()

```
_binding = FragmentDetailBinding.inflate(inflater,
container, false)
```

Digunakan untuk meng-inflate layout fragment menggunakan view binding. `Lau, binding.root` akan dikembalikan sebagai tampilan utama fragment.

c) Menerima Argument dari Fragment Lain

```
val title = arguments?.getString("TITLE")
val photo = arguments?.getString("PHOTO")
val singer = arguments?.getString("SINGER")
val album = arguments?.getString("ALBUM")
val lyrics = arguments?.getString("LYRICS")
```

Digunakan untuk menerima data berupa String dari Fragment sebelumnya menggunakan Bundle.

d) Menampilkan Data ke UI

```
binding.songTitle.text = title
binding.songSinger.text = singer
binding.songAlbum.text = album
binding.songLyrics.text = lyrics
```

Menampilkan data song ke dalam TextView di layout.

e) Menampilkan Gambar dengan Glide

```
photo?.let {
    Glide.with(requireContext())
        .load(it)
        .into(binding.songCover)
}
```

Digunakan untuk menampilkan gambar dari URL ke dalam Image View songCover menggunakan library Glide.

f) Tombol Back di Toolbar

```
binding.toolbar.setNavigationOnClickListener {
    parentFragmentManager.popBackStack()
}
```

Digunakan untuk mengatur aksi tombol back di toolbar agar Kembali ke Fragment sebelumnya.

g) onDestroyView()

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

Digunakan untuk membersihkan objek binding saat fragment dihancurkan agar tidak terjadi memory leak.

4. HomeFragment.kt

File ini merupakan tampilan utama dari aplikasi yang menampilkan daftar song dalam bentuk RecyclerView. Terdapat beberapa bagian, yaitu:

a) View Binding pada Fragment

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

Digunakan untuk mengakses elemen XML (fragment_home.xml) secara langsung dan aman dari NullPointerException.

b) Deklarasi Variabel

```
private lateinit var songAdapter: ListSongAdapter
private val list = ArrayList<Song>()
```

Di sini terdapat adapter khusus untuk menampilkan data song dalam RecyclerView dan list digunakan untuk menyimpan data lagu dari resource (array).

c) onCreateView()

```
list.clear()
list.addAll(getListSong())
setupRecyclerView()
```

Digunakan untuk mengambil data dari array dan menyimpannya dala list. Kemudian, memanggil fungsi setupRecyclerView() untuk menampilkan daftar lagu ke layar.

d) setupRecyclerView()

```
songAdapter = ListSongAdapter(
    list,
    onSpotifyClick = { link -> ... },
    onDetailClick = { title, photo, ... -> ... }
)
```

onSpotifyClick: ketika tombol Spotify di item lagu ditekan, membuka link ke aplikasi/browser Spotify.

onDetailClick: ketika item lagu ditekan, pindah ke DetailFragment dan mengirim data lagu lewat arguments.

e) getListSong()

```

val dataTitle =
resources.getStringArray(R.array.data_title)
...

```

Digunakan untuk mengambil semua data lagu dari file strings.xml (berupa array resource). Kemudian, data dikonversi menjadi objek Song dan dimasukkan ke list.

f) Navigasi ke DetailFragment

```

val detailFragment = DetailFragment().apply {
    arguments = Bundle().apply {
        putString("TITLE", title)
        ...
    }
}

```

Digunakan untuk membuat instance detailFragment, lalu mengirim data melalui arguments. Kemudian, tampilan diubah dengan DetailFragment menggunakan FragmentTransaction.

g) onDestroyView()

```

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

```

Digunakan untuk membersihkan binding saat tampilan dihancurkan untuk mencegah memory leak

h) Fungsi Log

```

- Log.d("HomeFragment", "Spotify button clicked
for link: $link")

```

Berfungsi untuk menampilkan log ketika pengguna menekan tombol Spotify dan memastikan bahwa klik pada tombol Spotify terdeteksi dan link-nya benar.

```

- Log.e("Intent to Spotify", "Going to $link")

```


Berfungsi untuk menampilkan log error.

- `Log.d("HomeFragment", "Detail button clicked - Title: $title, Singer: $singer, Album: $album")`

Berfungsi untuk menampilkan informasi ketika pengguna menekan tombol Detail dan memastikan data yang dikirim ke DetailFragment sudah tepat.

5. HomeViewModel.kt

File ini merupakan kelas ViewModel yang berfungsi untuk mengelola dan menyediakan data Song ke HomeFragment. Di mana data diambil dari resources dan dikirimkan secara asynchronous menggunakan StateFlow.

a) StateFlow

```
private val _songList =  
    MutableStateFlow<List<Song>>(emptyList())  
val songList: StateFlow<List<Song>> get() =  
    _songList
```

_songList sebagai penyimpanan data lagu yang dapat berubah (mutable).

songList sebagai versi read-only yang digunakan fragment untuk mengobservasi perubahan data.

b) Function getSongFlow()

```
private fun getSongFlow(): Flow<List<Song>>
```

Digunakan untuk mengambil data dari strings.xml yang berupa array. Kemudian, data dikonversi menjadi list Song dan dikirim menggunakan `emit()` ke flow.

c) Function loadSongs()

```
fun loadSongs() {  
    viewModelScope.launch {  
        collectSongs()    }  
}
```

```

    }
}

```

Digunakan untuk memulai proses pemuatan lagu. Bagian ini menggunakan `coroutine viewModelScope` untuk menjalankan secara `lifecycle-aware`.

d) Function `collectSongs()`

```
private suspend fun collectSongs()
```

Digunakan untuk emngosongkan data terlebih dahulu (`onStart`) dan lalu mengisi ulang. Kemudian, hasil dari flow ke `_songList` disimpan.

`onDetailClick`: ketika item lagu ditekan, pindah ke `DetailFragment` dan mengirim data lagu lewat `arguments`.

e) Fungsi Log

```
Log.d("HomeViewModel", "Songs loaded:
${songs.size} items")
```

Digunakan untuk menampilkan jumlah lagu yang dimuat ke log, membantu debugging saat data diambil dari resource.

6. ListSongAdapter.kt

File ini adalah adapter untuk `RecyclerView` yang bertanggung jawab menampilkan daftar lagu dalam tampilan list. Adapter ini juga menangani event klik tombol Spotify dan tombol Detail untuk setiap item lagu. Terdapat beberapa bagian, yaitu:

a) Deklarasi Class dan Konstruktor

```
class ListSongAdapter(
    private val listSong: ArrayList<Song>,
    private val onSpotifyClick: (String) -> Unit,
    private val onDetailClick: (String, String,
String, String, String) -> Unit)
:
RecyclerView.Adapter<ListSongAdapter.ListViewHold
er>()
```

listSong merupakan dat berisi objek Song.

onSpotifyClick merupakan function yang dipanggil saat tombol Spotify diklik.

onDetailClick merupakan function saat tombol Detail diklik dan data lagu dikirim ke fragment detail.

b) ViewHolder

```
inner class ViewHolder(val binding:
ItemSongBinding) :
RecyclerView.ViewHolder(binding.root)
ViewHolder menyimpan referensi ke layout item lagu (item_song.xml)
menggunakan View Binding.
```

c) bind()

```
fun bind(song: Song) {
    binding.tvTitle.text = song.title
    binding.tvDesc.text = song.desc
    Glide.with(binding.root.context)
        .load(song.photo)
        .into(binding.imgCover)

    binding.buttonSpotify.setOnClickListener {
        onSpotifyClick(song.link)
    }

    binding.buttonDetail.setOnClickListener {
        onDetailClick(song.title, song.photo,
song.singer, song.album, song.lyrics)
    }
}
```

tvTitle dan tvDesc digunakan untuk menampilkan judul dan deskripsi song.

Glide digunakan untuk memuat dan menampilkan gambar cover song dari URL.

buttonSpotify digunakan untuk membuka link Spotify.

buttonDetail digunakan untuk mengirim data lagu ke DetailFragment.

d) onCreateViewHolder()

```
val binding = ItemSongBinding.inflate(LayoutInflater.from(parent.context), parent, false)
return ViewHolder(binding)
```

Bagian ini berfungsi membuat tampilan satu item daftar lagu menggunakan item_song.xml dan mengembalikan ViewHolder.

e) getItemCount()

```
override fun getItemCount(): Int = listSong.size
```

Bagian ini berfungsi untuk menentukan jumlah item dalam list, sesuai dengan jumlah data song.

f) onBindViewHolder

```
holder.bind(listSong[position])
```

Digunakan untuk memanggil fungsi bind() untuk menampilkan data song pada posisi yang sesuai di RecyclerView.

7. activity_main.xml

activity_main.xml adalah tata letak kosong dengan FrameLayout, dirancang agar fleksibel menampilkan konten fragment. Ini umum digunakan dalam aplikasi Android berbasis fragment agar UI dapat berganti tanpa membuat banyak activity.

8. Fragment_detail.xml

Terdapat beberapa bagian penting, yaitu:

a) Root Layout

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

ConstraintLayout digunakan sebagai root agar kita bisa menempatkan elemen UI secara fleksibel dengan constraint.

b) Toolbar

```
<androidx.appcompat.widget.Toolbar ... />
```

Digunakan untuk membuat custom toolbar.

c) NestedScrollView

```
<androidx.core.widget.NestedScrollView ... >
```

Digunakan untuk membungkus konten agar bisa di-scroll secara vertikal.

d) LinearLayout

Di dalam NestedScrollView, digunakan LinearLayout vertikal untuk menampilkan isi konten.

9. Fragment_home.kt

a) Root

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

Memungkinkan penempatan elemen UI secara fleksibel dengan constraint antar elemen.

b) Toolbar

```
<androidx.appcompat.widget.Toolbar ... />
```

Bagian ini dapat menampilkan judul halaman, yaitu “My Favourite Song”.

c) RecyclerView

```
<androidx.recyclerview.widget.RecyclerView ... />
```

Menampilkan list song secara vertikal (akan diatur melalui adapter di kode Kotlin).

10. item_song.xml

a) Root: CardView

```
<androidx.cardview.widget.CardView ...>
```

Merupakan pembungkus utama yang membentuk tampilan item seperti kartu dengan sudut melengkung (`cardCornerRadius="20dp"`).

b) ConstraintLayout

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

Digunakan untuk mengatur komponen di dalam CardView secara fleksibel.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya!

A. Pembahasan

Kelas `android.app.Application` adalah kelas dasar yang berfungsi menjaga status aplikasi secara global. Objek `Application` diinisialisasi pertama kali ketika proses aplikasi dijalankan, sehingga bersifat singleton sepanjang siklus hidup aplikasi. Dengan demikian kelas ini menyediakan `application context` yang konstan dan dapat diakses dari komponen mana pun (misal `Activity`, `Service`, `Receiver`) selama aplikasi aktif. Fungsi utamanya adalah bertindak sebagai entry point dan tempat sentral untuk mengelola sumber daya atau data bersama yang dipakai lintas komponen aplikasi, seperti menyimpan variabel global, konfigurasi, atau referensi objek yang dipakai bersama.

MODUL 5: CONNECTION TO THE INTERNET

SOAL 1

Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:

1. 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.
2. Gunakan KotlinX Serialization sebagai library JSON.
3. Gunakan library seperti Coil atau Glide untuk image loading.
4. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: <https://developer.themoviedb.org/docs/getting-started>
 - d. Log saat data item masuk ke dalam list
 - e. Log saat tombol Detail dan tombol Explicit Intent ditekan
 - f. Log data dari list yang dipilih ketika berpindah ke halaman Detail
5. Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll).
6. Gunakan caching strategy pada Room.
7. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

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

A. Source Code

1. MainActivity.kt

Tabel 26. Source Code MainActivity.kt Soal 1

1	package
2	com.example.movielist.presentation.ui.activity
3	
4	import android.content.Intent
5	import android.os.Bundle
6	import android.view.View
7	import android.widget.Toast
8	import androidx.activity.viewModels
9	import androidx.appcompat.app.AppCompatActivity
10	import androidx.appcompat.app.AppCompatActivity
11	import androidx.lifecycle.Observer
12	import
13	androidx.recyclerview.widget.LinearLayoutManager
14	import androidx.room.Room
15	import
16	com.example.movielist.data.local.MovieAppPreference
17	s
18	import
19	com.example.movielist.data.local.database.AppDatabase
20	se
21	import
22	com.example.movielist.data.remote.api.RetrofitClient
23	t
24	import
25	com.example.movielist.data.repository.MovieRepository
26	Impl
27	import
28	com.example.movielist.databinding.ActivityMainBinding
29	ng

```

30 import
31 com.example.movielist.domain.usecase.GetPopularMoviesUseCase
32
33 import
34 com.example.movielist.presentation.ui.adapter.MovieAdapter
35
36 import
37 com.example.movielist.presentation.viewmodel.MovieViewModel
38
39 import
40 com.example.movielist.presentation.viewmodel.ViewModelFactory
41
42 import com.example.movielist.utils.Result
43
44 class MainActivity : AppCompatActivity() {
45
46     private lateinit var binding:
47     ActivityMainBinding
48     private lateinit var movieAdapter: MovieAdapter
49     private lateinit var movieAppPreferences:
50     MovieAppPreferences
51
52     private val movieViewModel: MovieViewModel by
53     viewModels {
54         val apiService =
55         RetrofitClient.tmdbApiService
56         val database = Room.databaseBuilder(
57             applicationContext,
58             AppDatabase::class.java,
59             AppDatabase.DATABASE_NAME

```

60).build()
61	val movieDao = database.movieDao()
62	
63	val tmdbApiKey =
64	"efa2ab3869af63c9dd27712409e6737d"
65	movieAppPreferences.saveApiKey(tmdbApiKey)
66	
67	val movieRepositoryImpl =
68	MovieRepositoryImpl(apiService, movieDao,
69	tmdbApiKey)
70	val getPopularMoviesUseCase =
71	GetPopularMoviesUseCase(movieRepositoryImpl)
72	ViewModelFactory(getPopularMoviesUseCase)
73	}
74	
75	override fun onCreate(savedInstanceState:
76	Bundle?) {
77	super.onCreate(savedInstanceState)
78	binding =
79	ActivityMainBinding.inflate(layoutInflater)
80	setContentView(binding.root)
81	
82	movieAppPreferences =
83	MovieAppPreferences(this)
84	
85	setupRecyclerView()
86	observeViewModel()
87	setupDarkModeToggle()
88	
89	binding.btnRetry.setOnClickListener {

90	movieViewModel.fetchPopularMovies()
91	}
92	}
93	
94	private fun setupRecyclerView() {
95	movieAdapter = MovieAdapter()
96	binding.rvMovies.apply {
97	layoutManager =
98	LinearLayoutManager(this@MainActivity)
99	adapter = movieAdapter
100	}
101	
1102	movieAdapter.onItemClick = { movie ->
103	val intent = Intent(this,
104	DetailActivity::class.java).apply {
105	
106	putExtra(DetailActivity.EXTRA_MOVIE, movie)
107	}
108	startActivity(intent)
109	}
110	}
111	
112	private fun observeViewModel() {
113	movieViewModel.popularMovies.observe(this,
114	Observer { result ->
115	when (result) {
116	is Result.Loading -> {
117	binding.progressBar.visibility
118	= View.VISIBLE
119	

120		binding.tvError.visibility =
121	View.GONE	
122		binding.btnRetry.visibility =
123	View.GONE	
124		binding.rvMovies.visibility =
125	View.GONE	
126		}
127		is Result.Success -> {
128		binding.progressBar.visibility
129	= View.GONE	
130		binding.tvError.visibility =
131	View.GONE	
132		binding.btnRetry.visibility =
133	View.GONE	
134		binding.rvMovies.visibility =
135	View.VISIBLE	
136		
137	movieAdapter.submitList(result.data)	
138		}
139		is Result.Error -> {
140		binding.progressBar.visibility
141	= View.GONE	
142		binding.rvMovies.visibility =
143	View.GONE	
144		binding.tvError.visibility =
145	View.VISIBLE	
146		binding.btnRetry.visibility =
147	View.VISIBLE	
148		binding.tvError.text = "Error:
149		\${result.exception.message}"

```

150         Toast.makeText(this, "Error:
151         ${result.exception.message}",
152         Toast.LENGTH_LONG).show()
153     }
154 }
155 })
156 }
157
158     private fun setupDarkModeToggle() {
159         binding.switchDarkMode.isChecked =
160         movieAppPreferences.getDarkModeState()
161
162
163         applyTheme(movieAppPreferences.getDarkModeState())
164
165
166         binding.switchDarkMode.setOnCheckedChangeListener {
167         _, isChecked ->
168
169         movieAppPreferences.saveDarkModeState(isChecked)
170             applyTheme(isChecked)
171         }
172     }
173
174     private fun applyTheme(isDarkMode: Boolean) {
175         if (isDarkMode) {
176
177         AppCompatDelegate.setDefaultNightMode(AppCompatDele
178         gate.MODE_NIGHT_YES)
179         } else {

```

180	
181	AppCompatDelegate.setDefaultNightMode(AppCompatDele
182	gate.MODE_NIGHT_NO)
183	}
184	}
185	}

2. MovieDao.kt

Tabel 27. Source Code MovieDao.kt Soal 1

1	package com.example.movielist.data.local.dao
2	
3	import androidx.room.Dao
4	import androidx.room.Insert
5	import androidx.room.OnConflictStrategy
6	import androidx.room.Query
7	import
8	com.example.movielist.data.local.entities.MovieEntit
9	y
10	
11	@Dao
12	interface MovieDao {
13	@Insert(onConflict = OnConflictStrategy.REPLACE)
14	suspend fun insertAllMovies(movies:
15	List<MovieEntity>)
16	
17	@Query("SELECT * FROM movies ORDER BY popularity
18	DESC")
19	suspend fun getAllMovies(): List<MovieEntity>
20	
21	@Query("DELETE FROM movies")

22	suspend fun clearAllMovies()
23	}

3. AppDatabase.kt

Tabel 28. Source Code AppDatabase.kt Soal 1

1	package com.example.movielist.data.local.database
2	
3	import androidx.room.Database
4	import androidx.room.RoomDatabase
5	import com.example.movielist.data.local.dao.MovieDao
6	import
7	com.example.movielist.data.local.entities.MovieEntity
8	
9	
10	@Database(entities = [MovieEntity::class], version =
11	1, exportSchema = false)
12	abstract class AppDatabase : RoomDatabase() {
13	abstract fun movieDao(): MovieDao
14	
15	companion object {
16	const val DATABASE_NAME = "tmdb_app_db"
17	}
18	}

4. MovieAppPreferences.kt

Tabel 29. Source Code MovieAppPreferences.kt Soal 1

1	package com.example.movielist.data.local
2	
3	import android.content.Context
4	import android.content.SharedPreferences

5	
6	class MovieAppPreferences(context: Context) {
7	
8	private val sharedPreferences: SharedPreferences
9	=
10	
11	context.getSharedPreferences("tmdb_app_prefs",
12	Context.MODE_PRIVATE)
13	
14	companion object {
15	private const val KEY_API_KEY = "api_key"
16	private const val KEY_DARK_MODE =
17	"dark_mode"
18	}
19	
20	fun saveApiKey(apiKey: String) {
21	
22	sharedPreferences.edit().putString(KEY_API_KEY,
23	apiKey).apply()
24	}
25	
26	fun getApiKey(): String? {
27	return
28	sharedPreferences.getString(KEY_API_KEY, null)
29	}
30	
31	fun saveDarkModeState(isDarkMode: Boolean) {
32	
33	sharedPreferences.edit().putBoolean(KEY_DARK_MODE,
34	isDarkMode).apply()

35	}
36	
37	fun getDarkModeState(): Boolean {
38	return
39	sharedPreferences.getBoolean(KEY_DARK_MODE, false)
40	}
41	}

5. RetrofitClient.kt

Tabel 30. Source Code RetrofitClient.kt Soal 1

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

21	}
22	
23	private val okHttpClient: OkHttpClient by lazy {
24	val logging = HttpLoggingInterceptor()
25	
26	logging.setLevel(HttpLoggingInterceptor.Level.BODY)
27	
28	OkHttpClient.Builder()
29	.addInterceptor(logging)
30	.connectTimeout(30, TimeUnit.SECONDS)
31	.readTimeout(30, TimeUnit.SECONDS)
32	.writeTimeout(30, TimeUnit.SECONDS)
33	.build()
34	}
35	
36	val tmdbApiService: TmdbApiService by lazy {
37	Retrofit.Builder()
38	.baseUrl(BASE_URL)
39	.client(okHttpClient)
40	
41	.addConverterFactory(json.asConverterFactory("applica
42	tion/json".toMediaType()))
43	.build()
44	.create(TmdbApiService::class.java)
45	}
46	}

6. TmdbApiService.kt

Tabel 31. Source Code TmdbApiService.kt Soal 1

1	package com.example.movielist.data.remote.api
---	---

2	
3	import
4	com.example.movielist.data.remote.models.MovieListRe
5	sponse
6	import retrofit2.Response
7	import retrofit2.http.GET
8	import retrofit2.http.Query
9	
1	interface TmdbApiService {
0	
1	@GET("movie/popular")
1	suspend fun getPopularMovies(
1	@Query("api_key") apiKey: String,
2	@Query("language") language: String = "en-
1	US",
3	@Query("page") page: Int = 1
1): Response<MovieListResponse>
4	}
1	
5	
1	
6	
1	
7	
1	
8	

7. MovieDto.kt

Tabel 32. Source Code MovieDto.kt Soal 1

1	package com.example.movielist.data.remote.models
---	--

```

2
3 import kotlinx.serialization.SerialName
4 import kotlinx.serialization.Serializable
5
6 @Serializable
7 data class MovieDto(
8     val adult: Boolean,
9     @SerialName("backdrop_path")
10    val backdropPath: String?,
11    @SerialName("genre_ids")
12    val genreIds: List<Int>,
13    val id: Int,
14    @SerialName("original_language")
15    val originalLanguage: String,
16    @SerialName("original_title")
17    val originalTitle: String,
18    val overview: String,
19    val popularity: Double,
20    @SerialName("poster_path")
21    val posterPath: String?,
22    @SerialName("release_date")
23    val releaseDate: String,
24    val title: String,
25    val video: Boolean,
26    @SerialName("vote_average")
27    val voteAverage: Double,
28    @SerialName("vote_count")
29    val voteCount: Int
30 )

```

8. MovieDtoExtension.kt

Tabel 33. Source Code MovieDtoExtension.kt Soal 1

```
1 package com.example.movielist.data.remote.models
2
3 import com.example.movielist.domain.model.Movie
4 import
5 com.example.movielist.data.local.entities.MovieEntit
6 y
7
8 fun MovieDto.toDomainMovie(): Movie {
9     return Movie(
10         id = id,
11         title = title,
12         overview = overview,
13         posterPath = posterPath,
14         releaseDate = releaseDate,
15         voteAverage = voteAverage
16     )
17 }
18
19 fun MovieDto.toMovieEntity(): MovieEntity {
20     return MovieEntity(
21         id = id,
22         title = title,
23         overview = overview,
24         posterPath = posterPath,
25         releaseDate = releaseDate,
26         voteAverage = voteAverage,
27         popularity = popularity
28     )
```

29	}
----	---

9. MovieListResponse.kt

Tabel 34. Source Code MovieListResponse.kt Soal 1

1	package com.example.movielist.data.remote.models
2	
3	import kotlinx.serialization.SerialName
4	import kotlinx.serialization.Serializable
5	
6	@Serializable
7	data class MovieListResponse(
8	val page: Int,
9	val results: List<MovieDto>,
10	@SerialName("total_pages")
11	val totalPages: Int,
12	@SerialName("total_results")
13	val totalResults: Int
14)

10. MovieRepository.kt

Tabel 35. Source Code MovieRepository.kt Soal 1

1	package com.example.movielist.data.repository
2	
3	import com.example.movielist.data.local.dao.MovieDao
4	import
5	com.example.movielist.data.remote.api.TmdbApiService
6	import
7	com.example.movielist.data.remote.models.toDomainMovie
8	ie
9	

```

10 import
11 com.example.movielist.data.remote.models.toMovieEnti
12 ty
13 import com.example.movielist.domain.model.Movie
14 import com.example.movielist.utils.Result
15 import kotlinx.coroutines.flow.Flow
16 import kotlinx.coroutines.flow.flow
17 import retrofit2.HttpException
18 import java.io.IOException
19
20 interface MovieRepository {
21     fun                                getPopularMovies():
22     Flow<Result<List<Movie>>>
23 }
24
25 class MovieRepositoryImpl(
26     private val apiService: TmdbApiService,
27     private val movieDao: MovieDao,
28     private val apiKey: String
29 ) : MovieRepository {
30
31     override fun                                getPopularMovies():
32     Flow<Result<List<Movie>>> = flow {
33         emit(Result.Loading)
34
35         val                                cachedMovies =
36 movieDao.getAllMovies().map { it.toDomainMovie() }
37         if (cachedMovies.isNotEmpty()) {
38             emit(Result.Success(cachedMovies))
39         }

```


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

	}
	}

11. Movie.kt

Tabel 36. Source Code Movie.kt Soal 1

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

12. GetPopularMoviesUseCase.kt

Tabel 37. Source Code GetPopularMoviesUseCase.kt Soal 1

1	package com.example.movielist.domain.usecase
2	
3	import com.example.movielist.domain.model.Movie
4	import
5	com.example.movielist.data.repository.MovieRepository
6	Impl
7	import com.example.movielist.utils.Result
8	import kotlinx.coroutines.flow.Flow

9	
10	class GetPopularMoviesUseCase (
11	private val movieRepository: MovieRepositoryImpl
12) {
13	operator fun invoke(): Flow<Result<List<Movie>>>
14	{
15	return movieRepository.getPopularMovies()
16	}
17	}

13. DetailActivity.kt

Tabel 38. Source Code DetailActivity.kt Soal 1

1	package
2	com.example.movielist.presentation.ui.activity
3	
4	import android.os.Build
5	import android.os.Bundle
6	import android.view.MenuItem
7	import android.widget.Toast
8	import androidx.appcompat.app.AppCompatActivity
9	import com.bumptech.glide.Glide
10	import
11	com.example.movielist.databinding.ActivityDetailBind
12	ing
13	import com.example.movielist.domain.model.Movie
14	
15	class DetailActivity : AppCompatActivity() {
16	
17	private lateinit var binding:
18	ActivityDetailBinding

```

19
20     companion object {
21         const val EXTRA_MOVIE = "extra_movie"
22     }
23
24     override fun onCreate(savedInstanceState:
25 Bundle?) {
26         super.onCreate(savedInstanceState)
27         binding =
28 ActivityDetailBinding.inflate(layoutInflater)
29         setContentView(binding.root)
30
31
32
33 supportActionBar?.setDisplayHomeAsUpEnabled(true)
34
35
36         val movie = if (Build.VERSION.SDK_INT >=
37 Build.VERSION_CODES.TIRAMISU) {
38             intent.getParcelableExtra(EXTRA_MOVIE,
39 Movie::class.java)
40         } else {
41             @Suppress("DEPRECATION")
42             intent.getParcelableExtra(EXTRA_MOVIE)
43         }
44
45         movie?.let {
46
47             supportActionBar?.title = it.title
48

```

49	binding.apply {
50	tvDetailTitle.text = it.title
51	tvDetailReleaseDate.text = "Release
52	Date: \${it.releaseDate}"
53	tvDetailVoteAverage.text = "Rating:
54	\${String.format("%.1f", it.voteAverage)}"
55	tvDetailOverview.text = it.overview
56	
57	val imageUrl =
58	"https://image.tmdb.org/t/p/w500\${it.posterPath}"
59	Glide.with(this@DetailActivity)
60	.load(imageUrl)
61	.centerCrop()
62	.into(ivDetailPoster)
63	}
64	} ?: run {
65	Toast.makeText(this, "Film tidak
66	ditemukan.", Toast.LENGTH_SHORT).show()
67	finish()
68	}
69	}
70	
71	
72	override fun onOptionsItemSelected(item:
73	MenuItem): Boolean {
74	if (item.itemId == android.R.id.home) {
75	onBackPressedDispatcher.onBackPressed()
76	return true
77	}
78	return super.onOptionsItemSelected(item)

79	}
80	}
81	

14. MovieAdapter.kt

Tabel 39. Source Code MovieAdapter.kt Soal 1

1	package com.example.movielist.presentation.ui.adapter
2	
3	import android.view.LayoutInflater
4	import android.view.ViewGroup
5	import androidx.recyclerview.widget.DiffUtil
6	import androidx.recyclerview.widget.ListAdapter
7	import androidx.recyclerview.widget.RecyclerView
8	import com.bumptech.glide.Glide
9	import
10	com.example.movielist.databinding.ItemMovieBinding
11	import com.example.movielist.domain.model.Movie
12	
13	class MovieAdapter : ListAdapter<Movie,
14	MovieAdapter.MovieViewHolder>(MovieDiffCallback()) {
15	
16	var onItemClick: ((Movie) -> Unit)? = null
17	
18	override fun onCreateViewHolder(parent: ViewGroup,
19	viewType: Int): MovieViewHolder {
20	val binding =
21	ItemMovieBinding.inflate(LayoutInflater.from(parent.co
22	ntext), parent, false)
23	return MovieViewHolder(binding)
24	}

```

25
26     override fun onBindViewHolder(holder:
27 MovieViewHolder, position: Int) {
28         val movie = getItem(position)
29         holder.bind(movie)
30     }
31
32     inner class MovieViewHolder(private val binding:
33 ItemMovieBinding) :
34         RecyclerView.ViewHolder(binding.root) {
35
36         init {
37             binding.btnDetail.setOnClickListener {
38
39 onItemClick?.invoke(getItem(adapterPosition))
40             }
41         }
42
43         fun bind(movie: Movie) {
44             binding.apply {
45                 tvMovieTitle.text = movie.title
46                 tvReleaseDate.text = "Release Date:
47 ${movie.releaseDate}"
48                 tvVoteAverage.text = "Rating:
49 ${String.format("%.1f", movie.voteAverage)}"
50                 tvOverview.text = movie.overview
51
52                 val imageUrl =
53 "https://image.tmdb.org/t/p/w500${movie.posterPath}"
54

```

55	Glide.with(itemView.context)
56	.load(imageUrl)
57	.centerCrop()
58	.into(ivPoster)
59	}
60	}
61	}
62	
63	class MovieDiffCallback :
64	DiffUtil.ItemCallback<Movie>() {
65	override fun areItemsTheSame(oldItem: Movie,
66	newItem: Movie): Boolean {
67	return oldItem.id == newItem.id
68	}
69	
70	override fun areContentsTheSame(oldItem:
71	Movie, newItem: Movie): Boolean {
72	return oldItem == newItem
73	}
74	}
75	}
76	

15. MovieViewModel.kt

Tabel 40. Source Code MovieViewModel.kt Soal 1

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


```

6 import androidx.lifecycle.viewModelScope
7 import com.example.movielist.domain.model.Movie
8 import
9 com.example.movielist.domain.usecase.GetPopularMoviesUseCase
10
11 import com.example.movielist.utils.Result
12 import kotlinx.coroutines.launch
13
14 class MovieViewModel (
15     private val getPopularMoviesUseCase:
16     GetPopularMoviesUseCase
17 ) : ViewModel() {
18
19     private val _popularMovies =
20     MutableLiveData<Result<List<Movie>>>()
21     val popularMovies: LiveData<Result<List<Movie>>>
22     = _popularMovies
23
24     init {
25         fetchPopularMovies()
26     }
27
28     fun fetchPopularMovies() {
29         viewModelScope.launch {
30             getPopularMoviesUseCase().collect {
31 result ->
32                 _popularMovies.value = result
33             }
34         }
35     }

```

36	}
----	---

16. ViewModelFactory.kt

Tabel 41. Source Code ViewModelFactory.kt Soal 1

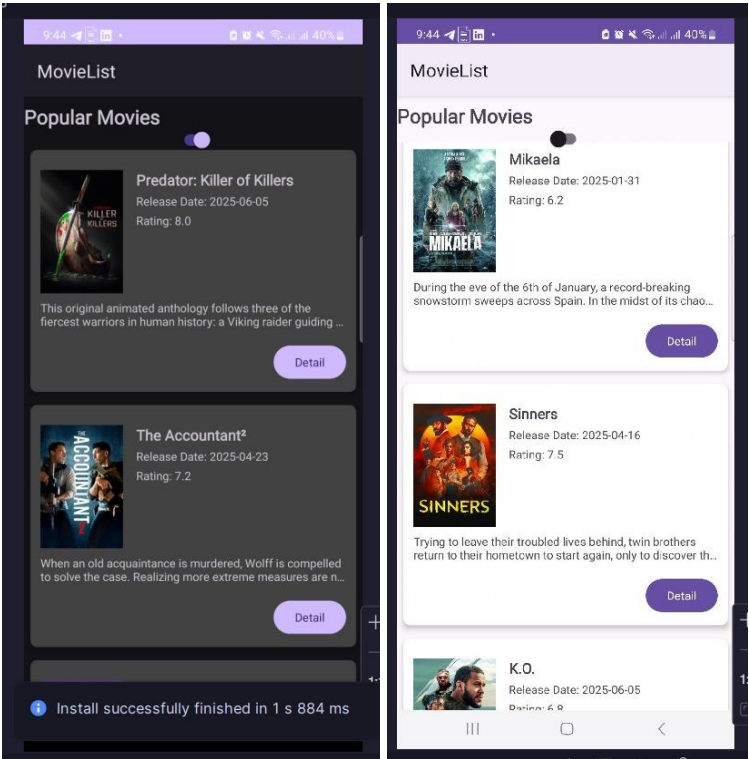
1	package com.example.movielist.presentation.viewmodel
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider
5	import
6	com.example.movielist.domain.usecase.GetPopularMoviesUseCase
7	
8	
9	class ViewModelFactory(
10	private val getPopularMoviesUseCase:
11	GetPopularMoviesUseCase
12) : ViewModelProvider.Factory {
13	
14	override fun <T : ViewModel> create(modelClass:
15	Class<T>): T {
16	if
17	(modelClass.isAssignableFrom(MovieViewModel::class.j
18	ava)) {
19	@Suppress("UNCHECKED_CAST")
20	return
21	MovieViewModel(getPopularMoviesUseCase) as T
22	}
23	throw IllegalArgumentException("Unknown
24	ViewModel class")
	}
	}

17. Result.kt

Tabel 42. Source Code Result Soal 1

1	package com.example.movielist.utils
2	
3	sealed class Result<out T> {
4	object Loading : Result<Nothing>()
5	data class Success<out T>(val data: T) :
6	Result<T>()
7	data class Error(val exception: Exception) :
8	Result<Nothing>()
9	}

B. Output Program



Gambar 10. Screenshot Hasil Soal 1

C. Pembahasan

1. MainActivity.kt

- **Inisialisasi ViewModel:**
Menggunakan viewModels dan ViewModelFactory untuk membuat MovieViewModel. MovieRepositoryImpl menggabungkan API dan Room database. API key disimpan via MovieAppPreferences.
- **onCreate():**
Inflate layout dengan ViewBinding (ActivityMainBinding). Inisialisasi MovieAppPreferences. Panggil fungsi untuk setup RecyclerView, ViewModel observer, dan Dark Mode. Tombol "Retry" untuk memuat ulang data jika gagal.
- **setupRecyclerView():**
Inisialisasi MovieAdapter dan pasang ke RecyclerView. Saat item diklik, buka DetailActivity dengan data film.
- **observeViewModel():**
Mengamati LiveData dari ViewModel (popularMovies). Tampilkan loading, data, atau error berdasarkan status (Result.Loading, Result.Success, Result.Error).
- **setupDarkModeToggle():**
Toggle switch untuk mengaktifkan/menonaktifkan Dark Mode. Status disimpan di MovieAppPreferences.
- **applyTheme():**
Menerapkan tema sesuai status Dark Mode (YES/NO).

2. MovieDao.kt

- a) **insertAllMovies(movies)**
Menyimpan daftar film ke database.
Jika ada konflik (misalnya ID sama), data lama akan diganti (REPLACE).
- b) **getAllMovies()**
Mengambil semua film dari tabel movies.
Diurutkan berdasarkan popularity secara menurun.
- c) **clearAllMovies()**

Menghapus seluruh data film dari tabel movies.

3. AppDatabase.kt

a) @Database(...)

Memberi tahu Room bahwa ini adalah database yang menyimpan data MovieEntity (yaitu data film).

version = 1: versi database-nya.

exportSchema = false: kita tidak perlu ekspor skema database ke file.

b) abstract fun movieDao(): MovieDao

Fungsi ini memberi akses ke DAO (Data Access Object) bernama MovieDao, yang isinya perintah buat masukan, ngambil, dan hapus data film.

c) companion object

Menyimpan nama database, yaitu "tmdb_app_db".

4. MovieAppPreferences.kt

a) saveApiKey(apiKey: String)

Menyimpan API key ke penyimpanan lokal (biar nggak perlu diketik ulang setiap kali buka aplikasi).

b) getApiKey()

Mengambil kembali API key yang sudah disimpan.

c) saveDarkModeState(isDarkMode: Boolean)

Menyimpan status Dark Mode (apakah aktif atau tidak).

5. RetrofitClient.kt

a) BASE_URL

Alamat dasar dari API TMDB:

"https://api.themoviedb.org/3/".

b) Json

Mengatur supaya parsing JSON dari API bisa mengabaikan data yang tidak dikenal dan tampil rapi saat di-log.

c) OkHttpClient

Dipakai oleh Retrofit untuk kirim-permintaan ke server.

Ada interceptor log untuk lihat request/response di Logcat.

Ada timeout 30 detik biar nggak nunggu selamanya kalau server lambat.

d) tmdbApiService

Objek yang dibuat Retrofit untuk mengakses fungsi-fungsi di

TmdbApiService.

Sudah siap dipakai di mana saja dalam aplikasi.

6. TmdbApiService.kt

a) @GET("movie/popular")

Menandakan bahwa ini adalah request GET ke endpoint movie/popular.

getPopularMovies(...)

Fungsi yang akan dipanggil untuk mengambil data film populer.

b) Parameter:

apiKey: kunci API TMDB (wajib).

language: bahasa hasil (default: "en-US").

c) page: halaman data (default: 1).

Response<MovieListResponse>

Data dari server akan dikembalikan dalam bentuk objek MovieListResponse, dibungkus dalam Response Retrofit.

7. MovieDto.kt

MovieDto adalah model data yang mencerminkan satu item film yang diterima dari API TMDB (dalam format JSON).

8. MovieDtoExtension.kt

File ini berisi fungsi ekstensi untuk mengubah MovieDto (data dari API) menjadi dua bentuk lain, yaitu Movie → untuk ditampilkan di UI dan MovieEntity → untuk disimpan di database lokal.

9. MovieListResponse.kt

- a) page: halaman saat ini dari data yang ditampilkan.
- b) results: daftar film dalam bentuk List<MovieDto>.
- c) totalPages: jumlah total halaman yang tersedia.
- d) totalResults: total seluruh film yang tersedia.

10. MovieRepository.kt

File ini mengatur pengambilan data film populer, baik dari API maupun database lokal, lalu mengirimkannya ke ViewModel dalam bentuk Flow<Result<List<Movie>>>.

11. Movie.kt

Movie adalah model data di lapisan domain, yaitu bentuk film yang digunakan di dalam logika aplikasi dan UI.

12. GetPopularMovieUseCase.kt

Merupakan komponen di arsitektur bersih (clean architecture) yang mengatur satu tugas spesifik, yaitu mengambil daftar film populer dari repository.

13. DetailActivity.kt

DetailActivity adalah halaman detail film yang muncul setelah pengguna klik salah satu item film di daftar utama.

14. MovieAdaptor.kt

MovieAdapter adalah adapter untuk RecyclerView yang menampilkan daftar film dalam bentuk item (card) satu per satu.

15. MovieViewModel.kt

MovieViewModel adalah penghubung antara UI (tampilan) dan data film. Ia bertugas mengambil data dari use case dan memberikannya ke UI.

16. ViewModelFactory.kt

ViewModelFactory adalah pembuat khusus untuk MovieViewModel. Karena MovieViewModel butuh parameter (use case), kita nggak bisa pakai cara biasa (ViewModelProvider(this)[...]). Nah, di sinilah ViewModelFactory membantu.

17. Result.kt

Result adalah cara aplikasi ini mengemas hasil dari proses yang bisa berhasil atau gagal (misalnya saat ambil data dari internet).

TAUTAN GIT

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/Noviana21/Pemrograman-Mobile>