

**LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE**



Oleh:

Muhammad Raihan

NIM. 2310817110008

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

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Muhammad Raihan

NIM : 2310817110008

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Raka Azwar
NIM. 2210817210012

Ir. Eka Setya Wijaya, S.T., M.Kom
NIP. 198205082008011010

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI.....	3
DAFTAR GAMBAR	5
DAFTAR TABEL.....	7
MODUL 1 : ANDROID BASIC WITH KOTLIN	10
SOAL 1	10
A. Source Code.....	12
B. Output Program.....	23
C. Pembahasan	24
MODUL 2 : ANDROID LAYOUT WITH COMPOSE.....	33
SOAL 1	33
A. Source Code.....	36
B. Output Program.....	54
C. Pembahasan	55
SOAL 2	63
MODUL 3 : BUILD A SCROLLABLE LIST.....	65
SOAL 1	65
A. Source Code.....	67
B. Output Program.....	136
C. Pembahasan	139
SOAL 2	153
MODUL 4 : VIEWMODEL AND DEBUGGING.....	154
SOAL 1	154

A. Source Code.....	155
B. Output Program.....	199
C. Pembahasan	206
SOAL 2	217
MODUL 5 : CONNECT TO THE INTERNET	218
SOAL 1	218
A. Source Code.....	218
B. Output Program.....	279
C. Pembahasan	280
Tautan Git.....	297

DAFTAR GAMBAR

Gambar 1. Tampilan Awal Aplikasi Soal 1 Modul 1	10
Gambar 2. Tampilan Dadu Setelah Di Roll Soal 1 Modul 1	11
Gambar 3. Tampilan Roll Dadu Double	12
Gambar 4. Screenshot Hasil Jawaban Soal 1 Versi Jetpack Compose	23
Gambar 5. Screenshot Hasil Jawaban Soal 1 Versi XML	24
Gambar 6. Tampilan Awal Aplikasi Soal 1	34
Gambar 7. Tampilan Pilihan Persentase Tip	35
Gambar 8. Tampilan Aplikasi Setelah Dijalankan	35
Gambar 9. Screenshot Hasil Jawaban Soal 1 Versi Jetpack Compose	54
Gambar 10. Screenshot Hasil Jawaban Soal 1 Versi XML	55
Gambar 11. Contoh UI List	66
Gambar 12. Contoh UI Detail	67
Gambar 13. Screenshot List Screen Soal 1 Versi Jetpack Compose	136
Gambar 14. Screenshot Detail Screen Soal 1 Versi Jetpack Compose	137
Gambar 15. Screenshot List Screen Soal 1 Versi XML	138
Gambar 16. Screenshot Detail Screen Soal 1 Versi XML	139
Gambar 17. Contoh Penggunaan Debugger	155
Gambar 18. Screenshot List Screen Soal 1 Versi Jetpack Compose	199
Gambar 19. Screenshot Detail Screen Soal 1 Versi Jetpack Compose	200
Gambar 20. Screenshot List Screen Soal 1 Versi XML	201
Gambar 21. Screenshot Detail Screen Soal 1 Versi XML	202
Gambar 22. Screenshot Log Saat Data Item Masuk Ke Dalam List Versi Compose	203
Gambar 23. Screenshot Log Saat Tombol Detail Dan Tombol Explicit Intent Ditekan Versi Compose	203
Gambar 24. Screenshot Log Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Versi Compose	204

Gambar 25. Screenshot Debugging dengan Tool Debugger di Android Studio Versi Compose.....	204
Gambar 26. Screenshot Log Saat Data Item Masuk Ke Dalam List Versi XML	205
Gambar 27. Screenshot Log Saat Tombol Detail Dan Tombol Explicit Intent Ditekan Versi XML	205
Gambar 28. Screenshot Log Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Versi XML	206
Gambar 29. Screenshot Debugging dengan Tool Debugger di Android Studio Versi XML	206
Gambar 30. Screenshot List Screen	279
Gambar 31. Screenshot Detail Screen.....	280

DAFTAR TABEL

Tabel 1. Source Code MainActivity.kt Soal 1 Modul 1 Jetpack Compose.....	12
Tabel 2. Source Code MainActivity.kt Soal 1 Modul 1 XML.....	18
Tabel 3. Source Code activity_main.xml Soal 1 Modul 1 XML	20
Tabel 4. Source Code MainActivity.kt Jawaban Soal 1 Jetpack Compose	36
Tabel 5. Source Code MainActivity.kt Jawaban Soal 1 Versi XML	45
Tabel 6. Source Code activity_main.xml Jawaban Soal 1 Versi XML	47
Tabel 7. Source Code Jetpack Compose MainActivity.kt Soal 1	67
Tabel 8. Source Code Jetpack Compose Card.kt Soal 1	69
Tabel 9. Source Code Jetpack Compose ListScreen.kt Soal 1	80
Tabel 10. Source Code Jetpack Compose Detail.kt Soal 1	84
Tabel 11. Source Code Jetpack Compose Routes.kt Soal 1	93
Tabel 12. Source Code Jetpack Compose KamenRider.kt Soal 1	93
Tabel 13. Source Code Jetpack Compose KamenRiderList.kt Soal 1	94
Tabel 14. Source Code XML MainActivity.kt.....	97
Tabel 15. Source Code XML MainAdapter.kt.....	98
Tabel 16. Source Code XML ListFragment.kt.....	100
Tabel 17. Source Code XML DetailFragment.kt	103
Tabel 18. Source Code XML KamenRider.kt.....	107
Tabel 19. Source Code XML KamenRiderList.kt	107
Tabel 20. Source Code XML activity_main.xml	111
Tabel 21. Source Code XML adapter_main.xml	112
Tabel 22. Source Code XML fragment_list.xml.....	120
Tabel 23. Source Code XML fragment_detail.xml.....	122
Tabel 24. Source Code XML fragment_detail.xml (landscape)	127
Tabel 25. Source Code XML main_graph.xml.....	134
Tabel 26. Source Code Jetpack Compose Card.kt Soal 1	155
Tabel 27. Source Code Jetpack Compose ListScreen.kt Soal 1	166

Tabel 28. Source Code Jetpack Compose DetailScreen.kt Soal 1	171
Tabel 29. Source Code Jetpack Compose RiderListViewModel.kt Soal 1	179
Tabel 30. Source Code Jetpack Compose ViewModel.kt Soal 1	182
Tabel 31. Source Code Jetpack Compose TimberApp.kt Soal 1	184
Tabel 32. Source Code XML MainAdapter.kt.....	184
Tabel 33. Source Code XML ListFragment.kt.....	187
Tabel 34. Source Code XML DetailFragment.kt	191
Tabel 35. Source Code XML RiderListViewModel.kt.....	194
Tabel 36. Source Code XML DetailViewModel.kt	196
Tabel 37. Source Code XML TimberApp.kt	198
Tabel 38. Source Code AppDatabase.kt	219
Tabel 39. Source Code MovieDao.kt.....	220
Tabel 40. Source Code MovieEntity.kt.....	221
Tabel 41. Source Code MovieApiModel.kt	222
Tabel 42. Source Code RetrofitClient.kt.....	223
Tabel 43. Source Code TmdbApiService.kt	225
Tabel 44. Source Code MovieRepositoryImpl.kt	226
Tabel 45. Source Code Mappers.kt.....	231
Tabel 46. Source Code Movie.kt.....	233
Tabel 47. Source Code MovieRepository.kt.....	234
Tabel 48. Source Code GetMovieDetailsUseCase.kt	234
Tabel 49. Source Code GetPopularMoviesUseCase.kt.....	235
Tabel 50. Source Code MovieDetailScreen.kt.....	236
Tabel 51. Source Code MovieItem.kt	247
Tabel 52. Source Code MovieListScreen.kt	255
Tabel 53. Source Code MovieDetailViewModel.kt.....	263
Tabel 54. Source Code MovieViewModel.kt	268
Tabel 55. Source Code AppNavigation.kt	273
Tabel 56. Source Code Routes.kt.....	274
Tabel 57. Source Code AppSettings.kt	274

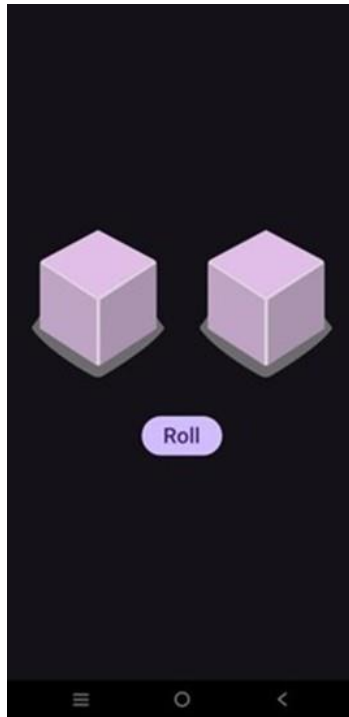
Tabel 58. Source Code NetworkResult.kt.....	276
Tabel 59. Source Code MainActivity.kt	277
Tabel 60. Source Code MyApplication.kt.....	278

MODUL 1 : ANDROID BASIC WITH KOTLIN

SOAL 1

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

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



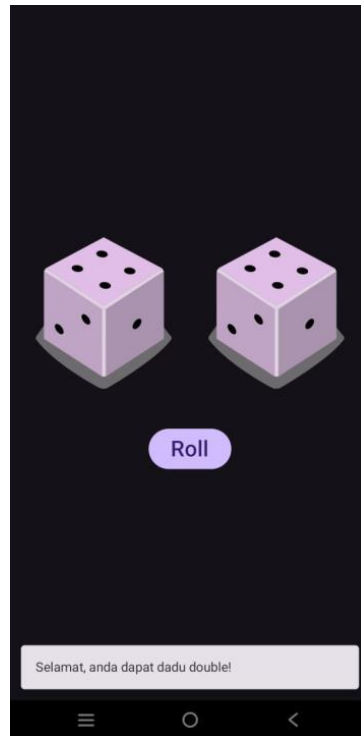
Gambar 1. Tampilan Awal Aplikasi Soal 1 Modul 1

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



Gambar 2. Tampilan Dadu Setelah Di Roll Soal 1 Modul 1

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 Roll Dadu Double

4. Buatlah aplikasi tersebut menggunakan XML dan Jetpack Compose.
5. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 1 dalam bentuk project**. Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repository.
6. Untuk gambar dadu dapat didownload pada link berikut:
https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export=download

A. Source Code

1) Versi Jetpack Compose

MainActivity.kt

Tabel 1. Source Code MainActivity.kt Soal 1 Modul 1 Jetpack Compose

1	package com.example.dicecompose
2	

3	import android.os.Bundle
4	import androidx.activity.ComponentActivity
5	import androidx.activity.compose.setContent
6	import androidx.activity.enableEdgeToEdge
7	import androidx.compose.foundation.Image
8	import
9	androidx.compose.foundation.layout.Arrangement
10	import androidx.compose.foundation.layout.Column
11	import androidx.compose.foundation.layout.Row
12	import androidx.compose.foundation.layout.Spacer
13	import
14	androidx.compose.foundation.layout.fillMaxSize
15	import androidx.compose.foundation.layout.height
16	import androidx.compose.foundation.layout.padding
	import androidx.compose.foundation.layout.size
17	import
18	androidx.compose.foundation.layout.wrapContentSize
19	import androidx.compose.material3.Button
20	import androidx.compose.material3.MaterialTheme
21	import androidx.compose.material3.Scaffold
22	import androidx.compose.material3.SnackbarDuration
23	import androidx.compose.material3.SnackbarHost
24	import androidx.compose.material3.SnackbarHostState
25	import androidx.compose.material3.Surface
26	import androidx.compose.material3.Text
27	import androidx.compose.runtime.Composable
28	import androidx.compose.runtime.getValue
29	import androidx.compose.runtime.mutableStateOf
	import androidx.compose.runtime.remember
30	

31	import	
32	androidx.compose.runtime.rememberCoroutineScope	
33	import androidx.compose.runtime.setValue	
34	import androidx.compose.ui.Alignment	
35	import androidx.compose.ui.Modifier	
36	import androidx.compose.ui.res.painterResource	
37	import androidx.compose.ui.res.stringResource	
38	import androidx.compose.ui.tooling.preview.Preview	
	import androidx.compose.ui.unit.dp	
39	import androidx.compose.ui.unit.sp	
40	import	
41	com.example.dicecompose.ui.theme.DiceComposeTheme	
42	import kotlinx.coroutines.launch	
43	class MainActivity : ComponentActivity() {	
44	override fun onCreate(savedInstanceState:	
45	Bundle?) {	
46	super.onCreate(savedInstanceState)	
47	enableEdgeToEdge()	
48	setContent {	
	DiceComposeTheme {	
49	Surface (
	modifier	=
50	Modifier.fillMaxSize(),	
51	color	=
52	MaterialTheme.colorScheme.background	
53) {	
54	DiceRollerApp()	
55	}	
56	}	

57	}
58	}
59	}
60	
61	@Preview(showBackground = true)
62	@Composable
63	fun DiceRollerApp() {
64	DiceWithButtonAndImage(modifier = Modifier
65	.fillMaxSize()
66	.wrapContentSize(Alignment.Center)
67)
68	}
69	@Composable
70	fun DiceWithButtonAndImage(modifier: Modifier =
71	Modifier) {
	var result1 by remember { mutableStateOf(0) }
72	var result2 by remember { mutableStateOf(0) }
73	val snackbarHostState = remember {
74	SnackbarHostState() }
75	val coroutineScope = rememberCoroutineScope()
76	val imageResource1 =
	getDiceImageResource(result1)
77	val imageResource2 =
78	getDiceImageResource(result2)
79	
	Scaffold (
80	

81	snackbarHost	=	{
82	SnackbarHost(snackbarHostState) },		
83	content = {padding ->		
84	Column(
	modifier = modifier		
85	.padding(padding),		
	verticalArrangement	=	
86	Arrangement.Center,		
87	horizontalAlignment	=	
88	Alignment.CenterHorizontally		
89) {		
	Row {		
90	Image(
	modifier	=	
91	Modifier.size(200.dp),		
	painter	=	
92	painterResource(imageResource1),		
93	contentDescription	=	
94	result1.toString()		
)		
95	Image(
	modifier	=	
96	Modifier.size(200.dp),		
	painter	=	
97	painterResource(imageResource2),		
98	contentDescription	=	
99	result2.toString()		
)		
100	}		
101			

102	Spacer(modifier	=
103	Modifier.height(16.dp))	
104	Button(onClick = {	
	result1 = (1..6).random()	
105	result2 = (1..6).random()	
106	coroutineScope.launch {	
107	snackbarHostState.showSnackbar(
	message	=
108	getSnackbarText(result1, result2),	
109	duration	=
110	SnackbarDuration.Short	
)	
111	}	
112)) {	
113		
114	Text(stringResource(R.string.roll), fontSize	=
115	20.sp)	
116	}	
117	}	
118	}	
119)	
120	}	
121		
122	fun getDiceImageResource (result: Int): Int {	
123	return when (result) {	
124	1 -> R.drawable.dice_1	
125	2 -> R.drawable.dice_2	
126	3 -> R.drawable.dice_3	

127	4 -> R.drawable.dice_4
128	5 -> R.drawable.dice_5
129	6 -> R.drawable.dice_6
130	else -> R.drawable.dice_0
	}
131	}
132	fun getSnackbarText (result1: Int, result2: Int):
133	String {
	return if(result1 == result2) "Selamat, Anda
	mendapatkan angka double!"
	else "Anda belum beruntung!"
	}

2) Versi XML

MainActivity.kt

Tabel 2. Source Code MainActivity.kt Soal 1 Modul 1 XML

1	package com.example.dicexml
2	
3	import android.os.Bundle
4	import android.widget.Button
5	import android.widget.ImageView
6	import androidx.activity.enableEdgeToEdge
7	import androidx.appcompat.app.AppCompatActivity
8	import
	androidx.constraintlayout.widget.ConstraintLayout
9	import com.google.android.material.snackbar.Snackbar
10	
11	class MainActivity : AppCompatActivity() {

12	override fun onCreate(savedInstanceState: Bundle?) {	
13	super.onCreate(savedInstanceState)	
14	enableEdgeToEdge()	
15	setContentView(R.layout.activity_main)	
16	val mainLayout: ConstraintLayout =	
	findViewById(R.id.main)	
17	val dice1: ImageView =	
	findViewById(R.id.dice1)	
18	val dice2: ImageView =	
	findViewById(R.id.dice2)	
19	val rollBtn: Button =	
	findViewById(R.id.rollBtn)	
20		
21	rollBtn.setOnClickListener {	
22	val result1 = (1..6).random()	
23	val result2 = (1..6).random()	
24		
25	val imageResource1 =	
	getDiceImageResource(result1)	
26	val imageResource2 =	
	getDiceImageResource(result2)	
27		
28	dice1.setImageResource(imageResource1)	
29	dice2.setImageResource(imageResource2)	
30		
31	val snackbarText =	
	getSnackbarText(result1, result2)	
32	Snackbar.make(mainLayout, snackbarText,	
	Snackbar.LENGTH_SHORT).show()	

33	}
34	}
35	}
36	
37	fun getDiceImageResource (result: Int): Int {
38	return when (result) {
39	1 -> R.drawable.dice_1
40	2 -> R.drawable.dice_2
41	3 -> R.drawable.dice_3
42	4 -> R.drawable.dice_4
43	5 -> R.drawable.dice_5
44	6 -> R.drawable.dice_6
45	else -> R.drawable.dice_0
46	}
47	}
48	
49	fun getSnackbarText (result1: Int, result2: Int):
	String {
50	return if(result1 == result2) "Selamat, Anda
	mendapatkan angka double!"
51	else "Anda belum beruntung!"
52	}

activity_main.xml

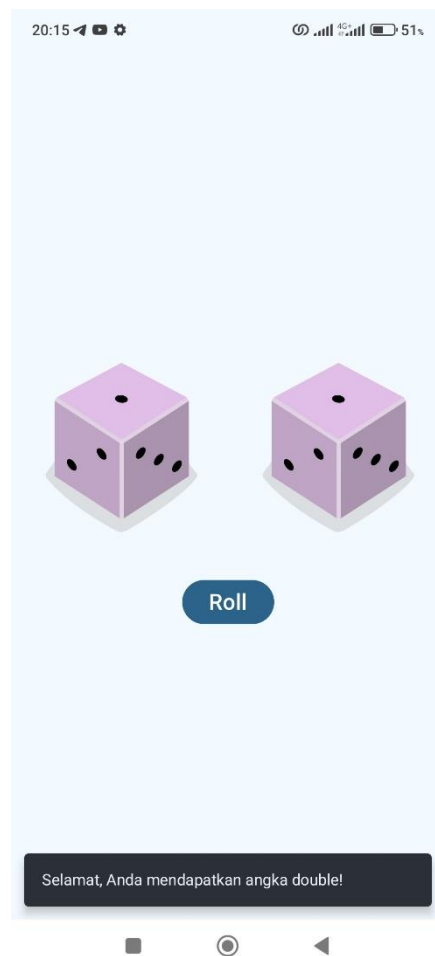
Tabel 3. Source Code activity_main.xml Soal 1 Modul 1 XML

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
	xmlns:android="http://schemas.android.com/apk/res/
	android"

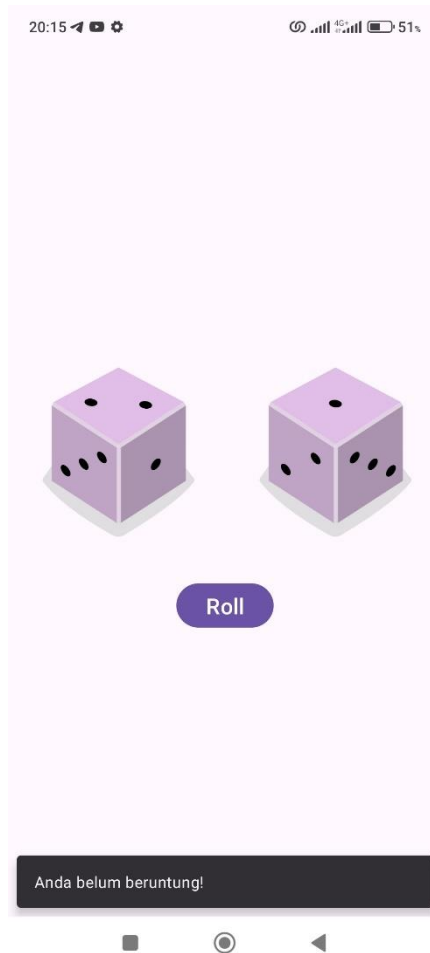
3	xmlns:app="http://schemas.android.com/apk/res-
	auto"
4	xmlns:tools="http://schemas.android.com/tools"
5	android:id="@+id/main"
6	android:layout_width="match_parent"
7	android:layout_height="match_parent"
8	tools:context=".MainActivity">
9	
10	<LinearLayout
11	android:layout_width="match_parent"
12	android:layout_height="match_parent"
13	android:orientation="vertical"
14	android:gravity="center">
15	
16	<LinearLayout
17	android:layout_width="match_parent"
18	android:layout_height="wrap_content"
19	app:layout_constraintTop_toTopOf="parent"
20	app:layout_constraintBottom_toBottomOf="parent"
21	android:orientation="horizontal"
22	android:gravity="center"
23	android:layout_marginBottom="15dp">
24	
25	<ImageView
26	android:id="@+id/dice1"
27	
28	android:layout_width="wrap_content"
29	android:layout_height="200dp"

30	android:src="@drawable/dice_0"
31	android:contentDescription="@string/dice1"/>
32	<ImageView
33	android:id="@+id/dice2"
34	
35	android:layout_width="wrap_content"
	android:layout_height="200dp"
36	android:src="@drawable/dice_0"
37	
38	android:contentDescription="@string/dice2"/>
39	</LinearLayout>
40	
41	<Button
42	android:id="@+id/rollBtn"
43	android:layout_width="wrap_content"
44	android:layout_height="wrap_content"
45	android:text="@string/roll"
46	android:textSize="20sp" />
47	</LinearLayout>
48	
49	
	</androidx.constraintlayout.widget.ConstraintLayout>

B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 Versi Jetpack Compose



Gambar 5. Screenshot Hasil Jawaban Soal 1 Versi XML

C. Pembahasan

1) Versi Jetpack Compose

MainActivity.kt

Pada baris [1], terdapat nama package file Kotlin yang dideklarasikan

Pada baris [3] sampai [39], terdapat beberapa komponen yang diimpor untuk kebutuhan pembuatan aplikasi.

Pada baris [41], terdapat class MainActivity yang dibuat, dimana class tersebut mewarisi dari `ComponentActivity()` yang merupakan activity yang mendukung Jetpack Compose.

Pada baris [42], terdapat fungsi `onCreate()` yang di-override, dimana fungsi tersebut akan dipanggil saat activity dimulai.

Pada baris [43], terdapat `onCreate()` yang dipanggil dari superclass-nya.

Pada baris [44], terdapat fungsi `enableEdgeToEdge()` yang dipanggil, dimana fungsi ini berguna untuk membuat tampilan layar aplikasi bisa menggunakan seluruh area layar.

Pada baris [45], terdapat fungsi `setContent` yang merupakan fungsi khusus dari Jetpack Compose, dimana fungsi ini berfungsi untuk menetapkan isi UI yang akan ditampilkan.

Pada baris [46], terdapat fungsi `DiceComposeTheme` yang berfungsi untuk menerapkan tema aplikasi seperti warna, tipografi, dan bentuk komponen pada seluruh UI.

Pada baris [47], terdapat sebuah container dengan nama `Surface` yang digunakan sebagai latar belakang untuk mengisi layar penuh dengan modifier `= Modifier.fillMaxSize()` dan mengatur warna latar belakang dengan `color = MaterialTheme.colorScheme.background`.

Pada baris [51], terdapat fungsi `DiceRollerApp()` yang dijalankan di dalam `Surface`, dimana fungsi ini merupakan fungsi yang akan menampilkan UI utama aplikasi.

Pada baris [60], terdapat fungsi `DiceRollerApp()` yang dibuat dan diberi anotasi `@Preview(showBackground = true)` untuk menampilkan preview UI aplikasi dengan tampilan latar belakangnya, serta anotasi `@Composable` yang menyatakan bahwa fungsi ini merupakan fungsi yang akan menampilkan UI di Compose.

Pada baris [61], terdapat fungsi `DiceWithButtonAndImage()` yang dijalankan di dalam `DiceRollerApp()`. Disini fungsi ini diberi argumen modifier `= Modifier.fillMaxSize()` yang dimana berfungsi untuk mengisi seluruh ukuran layar dan `wrapContentSize(Alignment.Center)` yang berfungsi untuk membuat isi di dalamnya ditaruh di tengah layar. Fungsi `DiceWithButtonAndImage()` berfungsi

untuk menampilkan gambar dadu dan tombol, serta mengatur logikanya saat ditampilkan di layar.

Pada baris [68], terdapat fungsi `DiceWithButtonAndImage()` yang dibuat, dimana fungsi ini berfungsi untuk membentuk tampilan dan logika dari UI utama. Fungsi ini akan menampilkan dua buah dadu, sebuah tombol untuk mengacak dadu, dan sebuah snackbar saat tombol ditekan. Fungsi ini juga memiliki parameter modifier yang bertipe data `Modifier` dengan nilai default `Modifier` kosong. Fungsi ini juga menggunakan anotasi `@Composable` yang menyatakan bahwa fungsi ini merupakan fungsi yang akan menampilkan UI di `Compose`.

Pada baris [69] dan [70], terdapat variabel `result1` dan `result2` yang berfungsi untuk menyimpan nilai dadu dari 1-6. Adapun statement `remember` pada kedua variabel tersebut berfungsi agar `Compose` tetap mengingat variabel meskipun terjadi perubahan pada UI. Statement `mutableStateOf(0)` mendefinisikan nilai awalnya adalah 0.

Pada baris [72], terdapat variabel `snackbarHostState` yang akan digunakan untuk mengatur dan menampilkan `Snackbar`.

Pada baris [73], terdapat variabel `coroutineScope` yang akan diperlukan untuk menjalankan fungsi `showSnackbar()`, dikarenakan fungsi ini merupakan fungsi `suspend` yang dimana harus ada `coroutineScope` untuk menjalankannya.

Pada baris [75] dan [76], terdapat variabel `imageResource1` dan `imageResource2` yang berfungsi untuk menyimpan resource gambar berdasarkan angka dadu dari `result1` dan `result2`. Variabel `imageResource1` akan menyimpan gambar berdasarkan hasil dari fungsi `getDiceImageResource()` dengan argumen `result1` dan variabel `imageResource2` akan menyimpan gambar berdasarkan hasil dari fungsi `getDiceImageResource()` dengan argumen `result2`.

Pada baris [78], terdapat `Scaffold` yang merupakan layout yang akan memberikan area untuk `Snackbar` dan konten utama. Kode di dalamnya dimulai dari baris [78] sampai baris [115].

Pada baris [79], terdapat `snackbarHost` yang didefinisikan berdasarkan `snackbarHostState` yang telah dibuat sebelumnya.

Pada baris [80], terdapat `content` yang merupakan tempat untuk konten utama yang akan berisi gambar dadu dan `button`.

Pada baris [81], terdapat `Column` yang akan menumpuk item secara vertikal. Item yang akan ditumpuk secara vertikal antara lain adalah `Row`, `Spacer`, dan `Button`. `Column` ini juga diberi argumen agar menggunakan `padding` dari `content` untuk menghindari `overlap`, membuat posisinya rata tengah secara vertikal dengan `verticalArrangement = Arrangement.Center`, dan membuat posisinya rata tengah secara horizontal dengan `horizontalAlignment = Alignment.CenterHorizontally`.

Pada baris [87], terdapat `Row` yang akan menumpuk item secara horizontal. `Row` disini berfungsi untuk menampilkan 2 buah item `Image` dadu secara horizontal.

Pada baris [88], terdapat `Image` yang `resourcenya` mengambil dari variabel `imageResource1`, ini akan menampilkan dadu pertama yang berada di sebelah kiri. `Image` ini diatur `size-nya` sebesar 200 dp.

Pada baris [93], terdapat `Image` yang `resourcenya` mengambil dari variabel `imageResource2`, ini akan menampilkan dadu kedua yang berada di sebelah kanan. `Image` ini diatur `size-nya` sebesar 200 dp.

Pada baris [99], terdapat `Spacer` yang berfungsi untuk memberi jarak antara `Row` dan `Button` setinggi 16 dp.

Pada baris [100], terdapat `Button` yang dibuat untuk mengacak dadu. Saat di klik, variabel `result1` dan `result2` akan diberi nilai acak dalam rentang angka 1-6. Lalu terdapat `coroutineScope.launch` untuk menjalankan fungsi `suspend showSnackbar` dari `snackbarHostState`. `Snackbar` akan menampilkan pesan yang didapat dari fungsi `getSnackbarText` yang diisi dengan argumen `result1` dan `result2` secara berurutan. Durasi `snackbar` saat tampil adalah `Short`.

Pada baris [111], terdapat Text dimana ini berfungsi untuk memberikan teks di dalam Button. Isi teks tersebut diambil dari resource file string.xml dengan nama “roll” yang isinya adalah Roll. Teks ini ukuran font-nya diatur menjadi sebesar 20 sp.

Pada baris [118], terdapat fungsi `getDiceImageResource()` yang dimana telah digunakan sebelumnya untuk mengambil resource gambar dadu berdasarkan result yang didapatkan saat menekan Button. Fungsi ini akan mengambil parameter result dan akan mengembalikan resource gambar dadu berdasarkan nilai dari result tersebut. Jika result bernilai 1, maka akan mengembalikan resource `dice_1`. Jika result bernilai 2, maka akan mengembalikan resource `dice_2`. Begitu seterusnya sampai result bernilai 6, maka akan mengembalikan resource `dice_6`. Jika result bernilai selain dari angka 1-6, maka akan mengembalikan resource `dice_0` yang dimana itu merupakan dadu kosong.

Pada baris [130], terdapat fungsi `getSnackbarText()` yang telah digunakan sebelumnya pada fungsi `showSnackbar` yang berfungsi untuk mengembalikan string yang akan ditampilkan sebagai pesan pada Snackbar. Fungsi ini memiliki parameter `result1` dan `result2` yang dimana jika nilai dari `result1` dan `result2` sama, maka akan mengembalikan string “Selamat, Anda mendapatkan angka double!”. Sedangkan jika tidak sama, maka akan mengembalikan string “Anda belum beruntung!”.

2) Versi XML

MainActivity.kt:

Pada baris [1], terdapat nama package file Kotlin yang dideklarasikan

Pada baris [3] sampai [9], terdapat beberapa komponen yang diimpor untuk kebutuhan pembuatan aplikasi.

Pada baris [11], terdapat class MainActivity yang dibuat, dimana class tersebut mewarisi dari AppCompatActivity() yang merupakan class bawaan Android yang menyediakan fitur activity modern dan digunakan pada project yang menggunakan XML.

Pada baris [12], terdapat fungsi onCreate() yang di-override, dimana fungsi tersebut akan dipanggil saat activity dimulai.

Pada baris [13], terdapat onCreate() yang dipanggil dari superclass-nya.

Pada baris [14], terdapat fungsi enableEdgeToEdge() yang dipanggil, dimana fungsi ini berguna untuk membuat tampilan layar aplikasi bisa menggunakan seluruh area layar.

Pada baris [15], terdapat fungsi setContentView yang berfungsi untuk mendeklarasikan layout pada activity_main.xml untuk ditampilkan ke layar.

Pada baris [16], terdapat variabel mainLayout yang dideklarasikan untuk menyimpan layout dari elemen ConstraintLayout dengan id 'main' yang berasal dari file xml.

Pada baris [17], terdapat variabel dice1 yang dideklarasikan untuk menyimpan resource gambar dari elemen ImageView dengan id 'dice1' yang berasal dari file xml.

Pada baris [18], terdapat variabel dice2 yang dideklarasikan untuk menyimpan resource gambar dari elemen ImageView dengan id 'dice2' yang berasal dari file xml.

Pada baris [19], terdapat variabel rollBtn yang dideklarasikan untuk menyimpan elemen Button dengan id 'rollBtn' yang berasal dari file xml.

Pada baris [21], terdapat variabel rollBtn yang diberi fungsi setOnClickListener untuk mengatur logika yang dilakukan ketika Button tersebut ditekan. Kode untuk logikanya terdapat pada baris [22] sampai [32].

Pada baris [22] dan [23], terdapat variabel `result1` dan `result2` yang dideklarasikan, dimana saat Button diklik maka akan mengisi kedua variabel ini dengan angka acak dalam rentang 1-6.

Pada baris [25] dan [26], terdapat variabel `imageResource1` dan `imageResource2` yang berfungsi untuk menyimpan resource gambar kedua dadu. Variabel `imageResource1` diisi dengan nilai yang didapat dari fungsi `getDiceImageResource` yang diisi dengan argumen `result1`. Sedangkan variabel `imageResource2` diisi dengan nilai yang didapat dari fungsi `getDiceImageResource` yang diisi dengan argumen `result2`.

Pada baris [28] dan [29], terdapat variabel `dice1` dan `dice2` dengan masing-masing memanggil fungsi `setImageResource()`. Variabel `dice1` memanggil fungsi `setImageResource()` dengan argumen `imageResource1`, yang artinya gambar pada `dice1` akan digantikan dengan gambar yang ada pada `imageResource1`. Sedangkan variabel `dice2` memanggil fungsi `setImageResource()` dengan argumen `imageResource2`, yang artinya gambar pada `dice2` akan digantikan dengan gambar yang ada pada `imageResource2`.

Pada baris [31], terdapat variabel `snackbarText` yang berfungsi untuk menyimpan isi teks atau pesan yang akan ditampilkan pada Snackbar. Isi dari variabel ini adalah hasil return dari fungsi `getSnackbarText` dengan argumen `result1` dan `result2`.

Pada baris [32], terdapat class `Snackbar` yang dipanggil dengan fungsi `make`. Di dalam fungsi `make` terdapat argumen `mainLayout` yang berfungsi sebagai context, `snackbarText` untuk teks atau pesan yang akan ditampilkan pada snackbar, dan `Snackbar.LENGTH_SHORT` untuk durasi Snackbar-nya. Selanjutnya langsung dipanggil fungsi `show()` untuk langsung menampilkan Snackbar pada layar.

Pada baris [37], terdapat fungsi `getDiceImageResource()` yang dimana telah digunakan sebelumnya untuk mengambil resource gambar dadu berdasarkan result yang didapatkan saat menekan Button. Fungsi ini akan mengambil parameter result dan akan mengembalikan resource gambar dadu berdasarkan nilai dari result

tersebut. Jika result bernilai 1, maka akan mengembalikan resource dice_1. Jika result bernilai 2, maka akan mengembalikan resource dice_2. Begitu seterusnya sampai result bernilai 6, maka akan mengembalikan resource dice_6. Jika result bernilai selain dari angka 1-6, maka akan mengembalikan resource dice_0 yang dimana itu merupakan dadu kosong.

Pada baris [49], terdapat fungsi `getSnackbarText()` yang telah digunakan sebelumnya saat membuat Snackbar yang berfungsi untuk mengembalikan string yang akan ditampilkan sebagai pesan pada Snackbar. Fungsi ini memiliki parameter `result1` dan `result2` yang dimana jika nilai dari `result1` dan `result2` sama, maka akan mengembalikan string “Selamat, Anda mendapatkan angka double!”. Sedangkan jika tidak sama, maka akan mengembalikan string “Anda belum beruntung!”.

activity_main.xml:

Pada baris [1], terdapat kode yang menjelaskan terkait versi XML dan encoding yang digunakan. Disini versinya adalah 1.0 dan encoding-nya adalah UTF-8.

Pada baris [2], terdapat `ConstraintLayout` yang menjadi dasar latar belakang tampilan aplikasi. Layout ini memiliki id ‘main’. Selain itu juga memiliki beberapa atribut seperti `layout_width=“match_parent”` agar lebarnya menjadi seluas layar, `layout_height=“match_parent”` agar tingginya menjadi seluas layar, dan `tools:context=“.MainActivity”` yang berarti akan digunakan sebagai preview di Android Studio.

Pada baris [10], terdapat `LinearLayout` bagian luar yang berfungsi untuk menampung 2 elemen di dalamnya, yaitu `LinearLayout` bagian dalam dan sebuah `Button`. Layout ini diatur lebar dan tingginya agar seluas layar. Lalu, layout ini juga diatur agar susunan orientasi elemen di dalamnya menjadi menumpuk secara vertikal. Selanjutnya, layout ini diatur dengan atribut `gravity=“center”` agar elemen-elemen di dalamnya menjadi rata tengah.

Pada baris [16], terdapat `LinearLayout` bagian dalam yang berfungsi untuk menampung 2 buah gambar dadu. Layout ini diatur lebarnya agar seluas layar dan tingginya menyesuaikan ukuran isi elemen di dalamnya. Posisi layout ini diatur sedemikian rupa dengan `ConstraintTop` dan `ConstraintBottom` agar posisinya menjadi rata tengah. Susunan orientasi elemen di dalam layout ini diatur menjadi horizontal. Lalu, layout ini diatur dengan atribut `gravity="center"` agar elemen dua dalamnya menjadi rata tengah. Selanjutnya, layout ini diberi `marginBottom` sebesar 15 dp untuk memberi jarak kepada `Button`.

Pada baris [25], terdapat `ImageView` dengan id `'dice1'` yang mengambil sumber gambarnya dari `dice_0` pada folder `drawable`. Gambar `dice_0` yang berupa dadu kosong diambil sebagai gambar default sebelum `Button` ditekan. Gambar ini diberi tinggi sebesar 200 dp dengan lebarnya menyesuaikan. `ContentDescription` pada gambar ini diambil dari string dengan nama `'dice1'` pada file `string.xml`.

Pada baris [31], terdapat `ImageView` dengan id `'dice2'` yang mengambil sumber gambarnya dari `dice_0` pada folder `drawable`. Gambar `dice_0` yang berupa dadu kosong diambil sebagai gambar default sebelum `Button` ditekan. Gambar ini diberi tinggi sebesar 200 dp dengan lebarnya menyesuaikan. `ContentDescription` pada gambar ini diambil dari string dengan nama `'dice2'` pada file `string.xml`.

Pada baris [39], terdapat `Button` dengan id `'rollBtn'` yang berisi teks dari string dengan nama `'roll'` dari file `string.xml`. Lebar dan tinggi `Button` ini menyesuaikan ukuran isi teks di dalamnya. Ukuran teks di dalam `Button` ini adalah sebesar 20 sp.


MODUL 2 : ANDROID LAYOUT WITH COMPOSE


SOAL 1

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

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

Calculate Tip

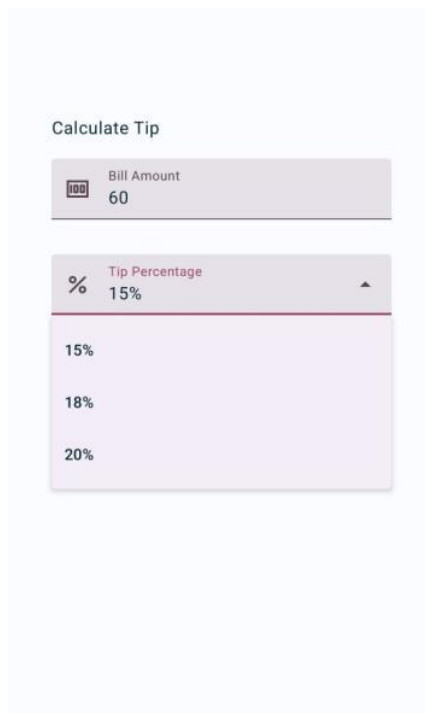
 Bill Amount

 Tip Percentage
15%

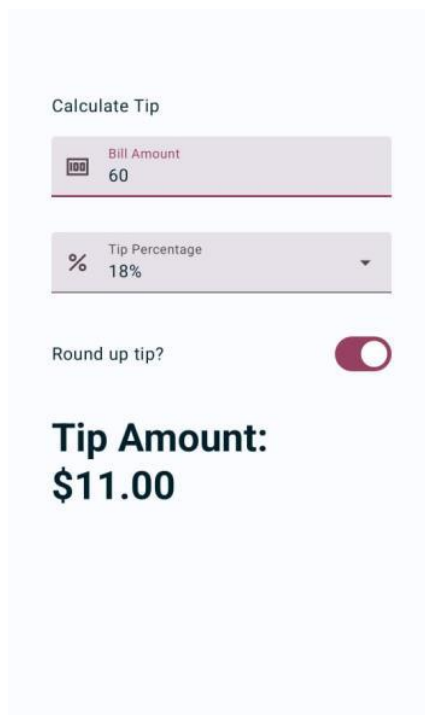
Round up tip? ☐

Tip Amount: \$0.00

Gambar 6. Tampilan Awal Aplikasi Soal 1



Gambar 7. Tampilan Pilihan Persentase Tip



Gambar 8. Tampilan Aplikasi Setelah Dijalankan

A. Source Code

1) Versi Jetpack Compose

MainActivity.kt

Tabel 4. Source Code MainActivity.kt Jawaban Soal 1 Jetpack Compose

1	package com.example.tipcalculatorcompose
2	
3	import android.os.Bundle
4	import androidx.activity.ComponentActivity
5	import androidx.activity.compose.setContent
6	import androidx.activity.enableEdgeToEdge
7	import androidx.annotation.DrawableRes
8	import androidx.annotation.StringRes
9	import
	androidx.compose.foundation.layout.Arrangement
10	import androidx.compose.foundation.layout.Column
11	import androidx.compose.foundation.layout.Row
12	import
	androidx.compose.foundation.layout.fillMaxSize
13	import
	androidx.compose.foundation.layout.fillMaxWidth
14	import androidx.compose.foundation.layout.height
15	import androidx.compose.foundation.layout.padding
16	import androidx.compose.foundation.layout.size
17	import
	androidx.compose.foundation.text.KeyboardOptions
18	import androidx.compose.material3.DropdownMenuItem
19	import
	androidx.compose.material3.ExperimentalMaterial3Ap
20	i

	import
21	androidx.compose.material3.ExposedDropDownMenuBox
	import
	androidx.compose.material3.ExposedDropDownMenuDefa
22	ults
23	import androidx.compose.material3.Icon
24	import androidx.compose.material3.Surface
25	import androidx.compose.material3.Switch
26	import androidx.compose.material3.Text
27	import androidx.compose.material3.TextField
28	import androidx.compose.runtime.Composable
29	import androidx.compose.runtime.getValue
30	import androidx.compose.runtime.mutableIntStateOf
31	import androidx.compose.runtime.mutableStateOf
32	import androidx.compose.runtime.remember
33	import androidx.compose.runtime.setValue
34	import androidx.compose.ui.Alignment
35	import androidx.compose.ui.Modifier
36	import androidx.compose.ui.res.painterResource
37	import androidx.compose.ui.res.stringResource
38	import androidx.compose.ui.text.font.FontWeight
39	import androidx.compose.ui.text.input.ImeAction
40	import androidx.compose.ui.text.input.KeyboardType
41	import androidx.compose.ui.tooling.preview.Preview
42	import androidx.compose.ui.unit.dp
43	import androidx.compose.ui.unit.sp
	import
	com.example.tipcalculatorcompose.ui.theme.TipCalcu
44	latorComposeTheme
45	import java.text.NumberFormat

```

46 import java.util.Locale
47
48 class MainActivity : ComponentActivity() {
49     override fun onCreate(savedInstanceState:
50 Bundle?) {
51         super.onCreate(savedInstanceState)
52         enableEdgeToEdge()
53         setContent {
54             TipCalculatorComposeTheme {
55                 Surface (
56                     Modifier.fillMaxSize()
57                 ) {
58                     TipTimeLayout()
59                 }
60             }
61         }
62     }
63 }
64
65 @OptIn(ExperimentalMaterial3Api::class)
66 @Composable
67 fun TipTimeLayout() {
68     val options = listOf(15, 18, 20)
69     var amountInput by remember { mutableStateOf("") }
70     var expanded by remember { mutableStateOf(false) }
71     var selectedOption by remember {
72 mutableIntStateOf(options[0]) }

```

73	val amount = amountInput.toDoubleOrNull() ?: 0.0
74	val tipPercent = selectedOption.toDouble()
75	var roundUp by remember { mutableStateOf(false)
	}
	val tipAmount = calculateTip(amount,
76	tipPercent, roundUp)
77	
78	Column (
79	modifier = Modifier
	.padding(horizontal = 40.dp),
80	horizontalAlignment =
	Alignment.CenterHorizontally,
81	verticalArrangement = Arrangement.Center
82) {
	Text (
83	text =
84	stringResource(R.string.calculate_tip),
85	modifier = Modifier
86	.padding(bottom = 15.dp)
87	.align(alignment = Alignment.Start)
88)
89	
90	EditNumberField(
91	label = R.string.bill_amount,
92	leadingIcon = R.drawable.money,
93	value = amountInput,
94	onValueChanged = { amountInput = it },
95	keyboardOptions = KeyboardOptions(
96	keyboardType = KeyboardType.Number,
97	imeAction = ImeAction.Next

98),
99	modifier = Modifier
100	.padding(bottom = 15.dp)
101	.fillMaxWidth()
102)
103	
104	ExposedDropDownMenuBox (
105	expanded = expanded,
106	onExpandedChange = {
107	expanded = !expanded
	},
108	modifier = Modifier.padding(bottom =
109	15.dp)
110) {
111	TextField(
112	modifier = Modifier
113	.menuAnchor()
114	.fillMaxWidth(),
115	readOnly = true,
116	value = "\$selectedOption%",
	onValueChange = { },
117	label = {
	Text(stringResource(R.string.tip)) },
118	leadingIcon = { Icon(painter =
119	painterResource(R.drawable.percent),
	contentDescription = null, Modifier.size(20.dp)) },
120	trailingIcon = {
121	
122	ExposedDropDownMenuDefaults.TrailingIcon(
123	expanded = expanded

124)
	},
125	colors =
126	ExposedDropDownMenuDefaults.textFieldColors()
127)
128	ExposedDropDownMenu(
129	expanded = expanded,
130	onDismissRequest = {
131	expanded = false
132	}
133) {
134	options.forEach { selectionOption -
	>
135	DropDownMenuItem(
136	text = { Text(text =
	"\$selectionOption%") },
137	onClick = {
138	selectedOption =
139	selectionOption
140	expanded = false
141	}
142)
143	}
144	}
145	}
146	
147	RoundTheTipRow(
	roundUp = roundUp,
148	onRoundUpChanged = { roundUp = it },
149	

150	modifier = Modifier.padding(bottom =
151	30.dp)
)
152	
153	Text(
154	text =
155	stringResource(R.string.tip_amount, tipAmount),
156	fontSize = 27.sp,
157	fontWeight = FontWeight.Bold
158)
159	}
160	}
161	
162	@Composable
163	fun EditNumberField(
164	value: String,
165	@StringRes label: Int,
166	@DrawableRes leadingIcon: Int,
167	onValueChanged: (String) -> Unit,
168	keyboardOptions: KeyboardOptions,
169	modifier: Modifier = Modifier
170) {
171	TextField(
172	value = value,
173	singleLine = true,
	modifier = modifier,
	onValueChange = onValueChanged,
174	label = {Text(stringResource(label))},
175	
176	

177	leadingIcon	=	{	Icon(painter	=
178	painterResource(id	=	leadingIcon),	null,	
179	Modifier.size(20.dp))	}			
180					
181	keyboardOptions	=	keyboardOptions		
182)				
183	}				
184	@Composable				
185	fun RoundTheTipRow(
186	roundUp: Boolean,				
187	onRoundUpChanged: (Boolean) -> Unit,				
188	modifier: Modifier = Modifier				
) {				
189	Row(
	modifier = modifier				
190	.fillMaxWidth()				
191	.height(48.dp),				
192	verticalAlignment	=			
	Alignment.CenterVertically,				
193	horizontalArrangement	=			
194	Arrangement.SpaceBetween				
195) {				
196	Text(
197	text	=			
198	stringResource(R.string.round_up_tip),				
199	modifier = Modifier.weight(1f)				
200)				
201	Switch(
202	checked = roundUp,				

	onCheckedChange = onRoundUpChanged
203)
204	}
205	}
206	
207	private fun calculateTip(amount: Double,
208	tipPercent: Double = 15.0, roundUp: Boolean): String
	{
	var tipAmount = (tipPercent / 100) * amount
209	if(roundUp){
210	tipAmount = kotlin.math.ceil(tipAmount)
211	}
212	
213	return
214	NumberFormat.getCurrencyInstance(Locale.US).format
215	(tipAmount)
216	}
217	
218	@Preview(
219	name = "Redmi Note 13",
220	widthDp = 360,
221	heightDp = 800,
222	showBackground = true
223)
	@Composable
	fun TipCalculatorPreview() {
	TipCalculatorComposeTheme {
	TipTimeLayout()
	}
	}

2) Versi XML

MainActivity.kt

Tabel 5. Source Code MainActivity.kt Jawaban Soal 1 Versi XML

1	package com.example.tipcalculatorxml
2	
3	import android.os.Bundle
4	import android.widget.AdapterView
5	import androidx.appcompat.app.AppCompatActivity
6	import androidx.core.widget.addTextChangedListener
7	import
	com.example.tipcalculatorxml.databinding.ActivityMa
	inBinding
8	import kotlin.math.ceil
9	
10	class MainActivity : AppCompatActivity() {
11	private lateinit var binding:
12	ActivityMainBinding
13	
	override fun onCreate(savedInstanceState:
14	Bundle?) {
15	super.onCreate(savedInstanceState)
	binding =
16	ActivityMainBinding.inflate(layoutInflater)
	setContentView(binding.root)
17	
18	val items = listOf("15%", "18%", "20%")
	val adapter = ArrayAdapter(this,
19	R.layout.list_items, items)

20	binding.dropdownField.setAdapter(adapter)	
	binding.dropdownField.setText(items[0],	
21	false)	
	binding.tipAmount.text	=
22	getString(R.string.tip_amount, 0.0)	
23		
24	fun updateTip(isRoundUp: Boolean) {	
	val billAmount	=
	binding.billAmountInput.text.toString().toDoubleOrNull()	
25	?: 0.0	
	val tipPercent	=
	binding.dropdownField.text.toString().replace("%",	
26	"").toDoubleOrNull() ?: 0.0	
	var tipAmount = billAmount * tipPercent	
27	/ 100	
	if(isRoundUp) { tipAmount	=
28	ceil(tipAmount) }	
	binding.tipAmount.text	=
29	getString(R.string.tip_amount, tipAmount)	
30	}	
31		
32	binding.billAmountInput.addTextChangedListener {	
33	updateTip(binding.switchRoundUp.isChecked)	
34	}	
	binding.dropdownField.addTextChangedListener {	
35		
36	updateTip(binding.switchRoundUp.isChecked)	

37	}
38	
	binding.switchRoundUp.setOnCheckedChangeListener {
39	_, isChecked ->
40	updateTip(isChecked)
41	}
42	
43	updateTip(false)
44	}
	}

activity_main.xml

Tabel 6. Source Code activity_main.xml Jawaban Soal 1 Versi XML

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
	xmlns:android="http://schemas.android.com/apk/res/
	android"
3	xmlns:app="http://schemas.android.com/apk/res-
	auto"
4	xmlns:tools="http://schemas.android.com/tools"
5	android:id="@+id/main"
6	android:layout_width="match_parent"
7	android:layout_height="match_parent"
8	android:padding="40dp"
9	tools:context=".MainActivity">
10	
11	<androidx.constraintlayout.widget.ConstraintLayout
12	android:layout_width="0dp"
13	android:layout_height="wrap_content"

14	app:layout_constraintTop_toTopOf="parent"
15	app:layout_constraintBottom_toBottomOf="parent"
16	app:layout_constraintStart_toStartOf="parent"
17	app:layout_constraintEnd_toEndOf="parent">
18	
19	<TextView
20	android:id="@+id/calc_tip"
21	android:layout_width="0dp"
22	android:layout_height="wrap_content"
23	
24	app:layout_constraintTop_toTopOf="parent"
25	app:layout_constraintStart_toStartOf="parent"
26	app:layout_constraintEnd_toEndOf="parent"
27	android:text="@string/calc_tip"
28	android:textSize="18sp"/>
29	<com.google.android.material.textfield.TextInputLayout
30	android:id="@+id/bill_amount"
31	android:layout_width="0dp"
32	android:layout_height="wrap_content"
33	android:layout_marginTop="15dp"
34	android:hint="@string/bill_amount"
35	android:minHeight="48dp"

36	app:layout_constraintEnd_toEndOf="parent"
37	app:layout_constraintStart_toStartOf="parent"
38	app:layout_constraintTop_toBottomOf="@id/calc_tip"
39	app:startIconDrawable="@drawable/money_resized">
40	
41	<com.google.android.material.textfield.TextInputEd itText
42	
43	android:id="@+id/bill_amount_input"
44	
45	android:layout_width="match_parent"
46	android:layout_height="wrap_content"
47	android:inputType="numberDecimal"
48	/>
49	</com.google.android.material.textfield.TextInputL ayout>
50	
51	<com.google.android.material.textfield.TextInputLa yout
52	android:id="@+id/dropdown_layout"
	android:layout_width="0dp"

53	app:layout_constraintTop_toBottomOf="@id/bill_amou
54	nt"
55	
56	app:layout_constraintStart_toStartOf="parent"
57	app:layout_constraintEnd_toEndOf="parent"
58	android:layout_height="wrap_content"
	android:layout_marginTop="15dp"
59	
	app:startIconDrawable="@drawable/percent_resized"
60	android:hint="@string/tip_percentage"
61	
62	style="@style/Widget.Material3.TextInputLayout.Out
63	linedBox.ExposedDropdownMenu">
64	
65	<AutoCompleteTextView
	android:id="@+id/dropdown_field"
66	
67	android:layout_width="match_parent"
68	android:layout_height="wrap_content"
	android:inputType="none"
69	
	tools:ignore="LabelFor,SpeakableTextPresentCheck"
70	/>
71	
72	
73	</com.google.android.material.textfield.TextInputL ayout>

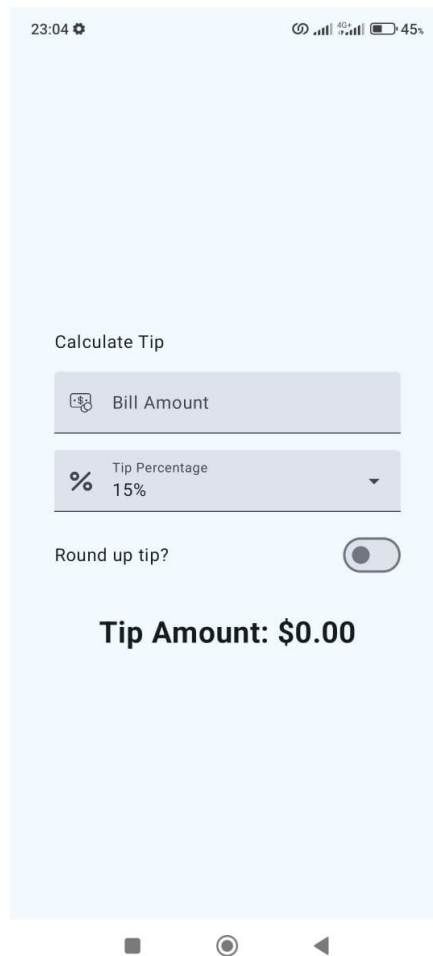
74	
75	<androidx.constraintlayout.widget.ConstraintLayout
76	android:id="@+id/roundUpLayout"
77	android:layout_width="0dp"
78	android:layout_height="wrap_content"
79	app:layout_constraintTop_toBottomOf="@id/dropdown_
80	layout"
81	app:layout_constraintStart_toStartOf="parent"
82	
83	app:layout_constraintEnd_toEndOf="parent"
84	android:layout_marginTop="10dp">
85	<TextView
86	android:id="@+id/roundUpTip"
87	android:layout_width="wrap_content"
88	android:layout_height="wrap_content"
89	app:layout_constraintTop_toTopOf="parent"
90	app:layout_constraintBottom_toBottomOf="parent"
91	app:layout_constraintStart_toStartOf="parent"
92	android:text="@string/round_up_tip"
93	android:textSize="16sp"/>
94	

95	<View
96	android:id="@+id/spacer"
	android:layout_width="0dp"
	android:layout_height="0dp"
97	
98	app:layout_constraintStart_toEndOf="@id/roundUpTip"
	"
99	app:layout_constraintTop_toTopOf="parent"
100	
101	app:layout_constraintBottom_toBottomOf="parent"
102	
	app:layout_constraintEnd_toStartOf="@id/switch_rou
103	nd_up"
104	app:layout_constraintHorizontal_weight="1"/>
105	
106	
107	<com.google.android.material.switchmaterial.Switch
108	Material
109	android:id="@+id/switch_round_up"
110	
111	android:layout_width="wrap_content"
112	android:layout_height="wrap_content"
113	
114	app:layout_constraintTop_toTopOf="parent"
115	
116	app:layout_constraintEnd_toEndOf="parent"

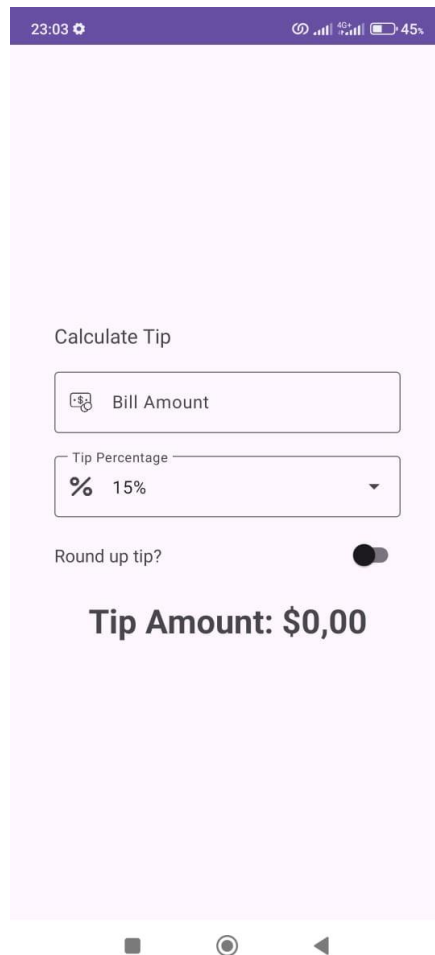
117	android:padding="0dp"
118	/>
119	
120	
121	</androidx.constraintlayout.widget.ConstraintLayout
122	t>
123	
124	<TextView
125	android:id="@+id/tip_amount"
126	android:layout_width="0dp"
127	android:layout_height="wrap_content"
128	app:layout_constraintTop_toBottomOf="@id/roundUpLayout"
	app:layout_constraintStart_toStartOf="parent"
	app:layout_constraintEnd_toEndOf="parent"
	android:layout_marginTop="15dp"
	android:textAlignment="center"
	android:textSize="30sp"
	android:textStyle="bold"/>
	</androidx.constraintlayout.widget.ConstraintLayout
	t>

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

B. Output Program



Gambar 9. Screenshot Hasil Jawaban Soal 1 Versi Jetpack Compose



Gambar 10. Screenshot Hasil Jawaban Soal 1 Versi XML

C. Pembahasan

1) Versi Jetpack Compose

MainActivity.kt

Pada baris pertama, terdapat deklarasi package `com.example.tipcalculatorcompose` yang menandai bahwa file ini berada dalam namespace `com.example.tipcalculatorcompose`. Ini penting dalam pengelompokan file dan penghindaran konflik nama class antar modul atau library.

Selanjutnya, baris-baris awal kode diisi dengan berbagai import statement yang digunakan untuk memanggil fungsi-fungsi dan komponen UI dari Android dan Jetpack Compose. Beberapa di antaranya adalah `androidx.compose.foundation.*` untuk elemen layout dasar, `androidx.compose.material3.*` untuk komponen Material Design versi 3, serta `androidx.compose.runtime.*` untuk mengelola state dalam Compose. Selain itu, terdapat juga import dari `java.text.NumberFormat` dan `java.util.Locale` yang digunakan untuk memformat hasil tip menjadi format mata uang.

Kelas utama aplikasi didefinisikan dengan class `MainActivity` : `ComponentActivity()` yang menandakan bahwa `MainActivity` adalah sebuah activity berbasis `ComponentActivity`. Di dalam metode `onCreate`, fungsi `enableEdgeToEdge()` dipanggil agar konten dapat ditampilkan penuh hingga ke tepi layar. Setelah itu, `setContent` digunakan untuk menampilkan UI dengan memanggil `TipTimeLayout()` di dalam tema `TipCalculatorComposeTheme`, dan seluruh UI dibungkus dalam `Surface` dengan ukuran memenuhi layar.

Fungsi `TipTimeLayout()` adalah fungsi `@Composable` yang berisi keseluruhan tampilan kalkulator tip. Fungsi ini ditandai dengan anotasi `@OptIn(ExperimentalMaterial3Api::class)` karena menggunakan komponen eksperimental seperti `ExposedDropDownMenuBox`. Di dalam fungsi ini, dibuat beberapa state: `amountInput` (menyimpan input jumlah tagihan dalam bentuk string), `expanded` (menyimpan status apakah dropdown menu terbuka atau tidak), dan `selectedOption` (menyimpan pilihan persen tip yang aktif). Nilai input dari pengguna dikonversi ke `Double` menggunakan `toDoubleOrNull()`, dan apabila gagal maka nilai default `0.0` akan digunakan. Nilai persen tip diambil dari `selectedOption`, dan `roundUp` adalah boolean yang menentukan apakah hasil tip perlu dibulatkan ke atas. Hasil akhir tip dihitung dengan memanggil `calculateTip()` dan disimpan dalam `tipAmount`.

Tampilan UI utama menggunakan `Column`, yaitu layout vertikal, dengan properti `horizontalAlignment` diatur ke tengah dan `verticalArrangement` ke tengah juga.

Komponen pertama di dalam kolom ini adalah Text yang menampilkan judul "Calculate Tip". Komponen ini diberi padding bawah dan disejajarkan ke kiri.

Setelah judul, terdapat input field yang dibangun dengan memanggil fungsi `EditNumberField`. Field ini menerima input angka dari pengguna sebagai jumlah tagihan, dilengkapi dengan label dan ikon uang di bagian kiri. Field ini hanya menerima satu baris teks, dengan jenis keyboard angka (`KeyboardType.Number`) dan aksi keyboard berikutnya (`ImeAction.Next`).

Komponen berikutnya adalah `ExposedDropDownMenuBox`, sebuah komponen untuk menampilkan dropdown menu Material3. Di dalamnya terdapat `TextField` yang hanya bisa dibaca (`readOnly`), menampilkan persen tip yang sedang dipilih. Bagian ikon kiri menampilkan ikon persen, dan bagian kanan menampilkan ikon panah dropdown. Saat dropdown terbuka (`expanded == true`), ditampilkan daftar opsi persen tip yang telah ditentukan (15%, 18%, dan 20%). Saat pengguna memilih salah satu, `selectedOption` akan diperbarui, dan dropdown akan tertutup.

Setelah dropdown, ditampilkan baris pengaturan untuk round up tip menggunakan fungsi `RoundTheTipRow`. Baris ini menampilkan teks "Round up tip?" di kiri dan Switch di kanan. Status Switch mengikuti nilai `roundUp` dan memperbarui nilai tersebut saat pengguna mengubahnya.

Terakhir, hasil perhitungan tip ditampilkan dengan Text besar dan tebal menggunakan `fontSize 27.sp` dan `fontWeight = FontWeight.Bold`. Nilai ini merupakan hasil format dari `calculateTip()`.

Fungsi `EditNumberField` adalah `@Composable` yang menerima string input, label, ikon, dan fungsi callback untuk perubahan nilai. Fungsi ini digunakan untuk membangun `TextField` yang dapat digunakan kembali di tempat lain jika dibutuhkan. Leading icon disisipkan menggunakan `painterResource` dengan ukuran 20.dp.

Fungsi `RoundTheTipRow` juga merupakan `@Composable` yang menampilkan baris dengan teks dan Switch. Komponen ini disusun horizontal dengan `Row`, di mana

teks di sisi kiri dan switch di sisi kanan. Tekan switch akan memicu `onRoundUpChanged` untuk memperbarui state `roundUp`.

Fungsi `calculateTip` adalah fungsi non-komposabel biasa yang menerima jumlah tagihan, persen tip, dan opsi pembulatan. Perhitungan dilakukan dengan mengalikan jumlah tagihan dan persen, lalu membulatkannya ke atas jika `roundUp` bernilai true menggunakan `kotlin.math.ceil()`. Hasil akhirnya diformat ke mata uang menggunakan `NumberFormat.getCurrencyInstance(Locale.US)`.

Terakhir, terdapat fungsi `TipCalculatorPreview` yang menggunakan anotasi `@Preview`. Fungsi ini membuat pratinjau tampilan aplikasi dengan ukuran layar menyerupai Redmi Note 13 (360x800 dp). Ini sangat membantu dalam proses desain UI agar dapat terlihat seperti tampil di perangkat nyata.

2) Versi XML

MainActivity.kt:

Kode dimulai dengan deklarasi package `com.example.tipcalculatorxml`, yang menunjukkan bahwa file ini termasuk dalam package bernama `com.example.tipcalculatorxml`. Ini merupakan cara umum dalam Android untuk mengorganisasi file sumber dan menghindari konflik nama antar class.

Setelah itu, berbagai import statement digunakan untuk memanggil class dan fungsi penting dari Android dan Kotlin. `android.os.Bundle` digunakan dalam siklus hidup activity. `android.widget.AdapterView` diperlukan untuk menyediakan daftar opsi pada dropdown. `androidx.appcompat.app.AppCompatActivity` adalah superclass untuk activity yang menggunakan kompatibilitas fitur lama pada Android. `androidx.core.widget.addTextChangedListener` menyediakan ekstensi Kotlin untuk mendengarkan perubahan teks secara lebih sederhana. `com.example.tipcalculatorxml.databinding.ActivityMainBinding` digunakan untuk mengakses elemen-elemen UI yang didefinisikan di file XML melalui View

Binding. Terakhir, `kotlin.math.ceil` digunakan untuk melakukan pembulatan ke atas pada hasil tip.

Di dalam kelas `MainActivity`, terdapat deklarasi `private lateinit var binding: ActivityMainBinding`. Ini adalah deklarasi `late-initialized property` yang memungkinkan kita untuk menginisialisasi binding nanti di `onCreate`, namun tetap bisa mengaksesnya sebagai properti.

Fungsi `onCreate` adalah titik masuk utama ketika activity dijalankan. Di dalamnya, binding diinisialisasi dengan memanggil `ActivityMainBinding.inflate(layoutInflater)`, yang akan mengikat elemen UI dari layout XML ke dalam objek Kotlin. Kemudian `setContentView(binding.root)` akan menampilkan layout tersebut sebagai tampilan activity.

Selanjutnya, dibuat sebuah daftar string berisi pilihan persen tip, yaitu "15%", "18%", dan "20%". Daftar ini kemudian digunakan untuk membuat `ArrayAdapter`, yang berfungsi sebagai adapter untuk dropdown `AutoCompleteTextView`. Adapter ini menggunakan layout `R.layout.list_items` sebagai tampilan dari masing-masing item dalam daftar. Dropdown (`binding.dropdownField`) kemudian diatur agar menggunakan adapter ini, dan secara default menampilkan pilihan pertama ("15%"). Label tip (`binding.tipAmount`) diatur agar menampilkan "Tip amount: \$0.0" sebagai nilai awal.

Sebuah fungsi lokal bernama `updateTip(isRoundUp: Boolean)` didefinisikan di dalam `onCreate`. Fungsi ini bertanggung jawab menghitung dan menampilkan tip berdasarkan input pengguna. Di dalam fungsi ini, teks dari `billAmountInput` dikonversi ke `Double`, jika gagal maka diasumsikan 0.0. Demikian juga dengan nilai persen tip yang diambil dari teks dropdown, dengan karakter % dibuang sebelum dikonversi menjadi angka desimal. Kemudian dilakukan perhitungan tip: $\text{billAmount} * \text{tipPercent} / 100$. Jika switch `round up` dalam keadaan aktif (`isRoundUp == true`), maka nilai tip dibulatkan ke atas dengan fungsi `ceil`. Terakhir, hasil tip diformat dan ditampilkan di `binding.tipAmount` menggunakan

getString(R.string.tip_amount, tipAmount) yang akan menghasilkan string seperti "Tip amount: \$2.0".

Untuk merespons perubahan pengguna, listener ditambahkan pada tiga komponen UI. Pertama, addTextChangedListener dipasang ke billAmountInput sehingga setiap perubahan angka tagihan akan menghitung ulang tip. Kedua, dropdownField juga memiliki listener yang akan memicu perhitungan ulang saat persen tip diubah. Ketiga, setOnCheckedChangeListener digunakan pada switch switchRoundUp, yang akan memanggil updateTip dengan nilai boolean baru setiap kali switch diubah.

Akhirnya, sebelum activity selesai diinisialisasi, updateTip(false) dipanggil untuk pertama kalinya agar hasil tip langsung dihitung berdasarkan kondisi awal aplikasi, tanpa harus menunggu interaksi pengguna.

activity_main.xml:

File XML ini menggunakan ConstraintLayout sebagai root layout, yaitu sebuah jenis layout yang fleksibel dan efisien dalam menyusun elemen UI dengan membatasi posisi relatif terhadap elemen lain. Root ConstraintLayout ini memiliki atribut lebar dan tinggi match_parent, yang berarti akan mengisi seluruh ruang yang tersedia di layar. Padding sebesar 40dp ditambahkan di sekeliling layout agar isi tidak menempel ke tepi layar. Atribut tools:context=".MainActivity" memberitahu Android Studio bahwa layout ini terkait dengan MainActivity.

Di dalam root layout, terdapat lagi sebuah ConstraintLayout yang menjadi pembungkus semua elemen UI utama. Layout ini diatur agar berada tepat di tengah parent-nya, dengan mengaitkan semua sisi (Top, Bottom, Start, dan End) ke sisi parent, dan lebar diatur 0dp agar menyesuaikan dengan constraint (match constraints).

Elemen pertama di dalam layout utama adalah sebuah TextView dengan ID `calc_tip`. Teks ini digunakan sebagai judul atau label utama dan menampilkan string dari resource `@string/calc_tip`. Lebar nya 0dp (menyesuaikan constraint), dengan tinggi `wrap_content`, dan teksnya berukuran 18sp.

Selanjutnya, ada sebuah TextInputLayout dengan ID `bill_amount` yang digunakan untuk membungkus kolom input nominal tagihan. Elemen ini menampilkan ikon uang (`@drawable/money_resized`) sebagai ikon awal dan memiliki hint yang berasal dari `@string/bill_amount`. Di dalamnya terdapat TextInputEditText dengan ID `bill_amount_input`, yang menerima input angka desimal (`numberDecimal`). Elemen ini penting karena menjadi dasar perhitungan tip nantinya.

Di bawah input nominal, terdapat TextInputLayout lain dengan ID `dropdown_layout`, yang berfungsi menampilkan pilihan persen tip. Layout ini menggunakan style `OutlinedBox.ExposedDropDownMenu`, yang secara otomatis membuat input terlihat seperti dropdown. Ikon awalnya adalah ikon persen (`@drawable/percent_resized`) dan hint-nya berasal dari `@string/tip_percentage`. Di dalamnya terdapat AutoCompleteTextView dengan ID `dropdown_field`, yang akan menampilkan daftar opsi persen tip yang bisa dipilih pengguna.

Setelah itu, terdapat ConstraintLayout bernama `roundUpLayout`, yang digunakan untuk meletakkan pengaturan pembulatan tip. Di dalamnya terdapat TextView dengan ID `roundUpTip` yang menampilkan label teks "Round up tip" dari string resource. Kemudian ada elemen View dengan ID `spacer` yang digunakan sebagai pemisah fleksibel untuk mendorong SwitchMaterial ke sisi kanan layout. Elemen SwitchMaterial dengan ID `switch_round_up` memungkinkan pengguna memilih apakah hasil tip akan dibulatkan ke atas atau tidak.

Terakhir, elemen TextView dengan ID `tip_amount` digunakan untuk menampilkan hasil perhitungan tip. Lebar nya 0dp agar mengisi ruang yang tersedia, tingginya `wrap_content`, dan posisinya berada di bawah pengaturan pembulatan. Teks ini

ditampilkan dengan ukuran besar 30sp, bold, dan rata tengah untuk menonjolkan hasil yang dihitung.

list_items.xml

File XML ini mendefinisikan sebuah elemen TextView yang memiliki ID textViewItem. Elemen TextView ini digunakan untuk menampilkan teks dalam suatu list item, yang kemudian bisa digunakan dalam elemen UI seperti AutoCompleteTextView atau ListView.

Atribut `android:layout_width="match_parent"` berarti lebar TextView akan mengisi seluruh lebar ruang yang tersedia dalam layout parent-nya. Atribut `android:layout_height="wrap_content"` mengindikasikan bahwa tinggi elemen ini akan menyesuaikan dengan konten teks yang ditampilkannya, sehingga hanya sebesar teks yang ditampilkan.

Atribut `android:padding="16dp"` memberikan jarak antara konten teks dan batas elemen TextView, memberikan ruang agar teks tidak menempel pada sisi elemen. Padding ini digunakan untuk membuat tampilan lebih rapi dan nyaman dibaca.

Atribut `android:textSize="18sp"` mengatur ukuran teks yang ditampilkan pada TextView. Ukuran font ini menggunakan satuan sp (scale-independent pixels), yang lebih fleksibel karena dapat menyesuaikan dengan preferensi ukuran teks yang diatur oleh pengguna pada perangkat mereka.

SOAL 2

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

Jawab:

1. Perbedaan Implementasi

Pada implementasi XML, struktur UI didefinisikan terlebih dahulu di file .xml, kemudian dikaitkan dengan logika dalam file Kotlin melalui View Binding. Misalnya, elemen-elemen seperti `TextInputEditText`, `AutoCompleteTextView`, `SwitchMaterial`, dan `TextView` ditulis dalam `activity_main.xml`, lalu diakses di `MainActivity.kt` menggunakan `binding.namaView`.

Sedangkan dalam Jetpack Compose, UI dan logika digabungkan menjadi satu alur deklaratif. Seluruh komponen UI—seperti inputan jumlah tagihan, dropdown persentase tip, switch pembulatan, dan hasil tip—langsung dibuat menggunakan fungsi-fungsi Kotlin seperti `TextField`, `ExposedDropdownMenuBox`, `Switch`, dan `Text`. Tidak ada file layout terpisah karena semua didefinisikan dalam kode Kotlin secara deklaratif.

2. Kelebihan dan Kekurangan

XML (View-based UI)

Kelebihan:

- Lebih familiar bagi developer Android lama karena sudah digunakan sejak awal Android.
- Visual Editor tersedia di Android Studio, memudahkan dalam menyusun UI secara drag-and-drop.

- Struktur lebih terpisah antara UI dan logic, yang bisa dianggap lebih rapi oleh beberapa developer.

Kekurangan:

- Membutuhkan boilerplate lebih banyak, seperti binding dan pengaturan listener manual.
- UI tidak reaktif secara otomatis terhadap perubahan state; semua pembaruan seperti perhitungan ulang tip harus dipanggil eksplisit pada setiap listener.
- Dropdown dan Switch butuh penanganan manual pada state-nya, tidak konsisten Compose dalam hal pengelolaan state.

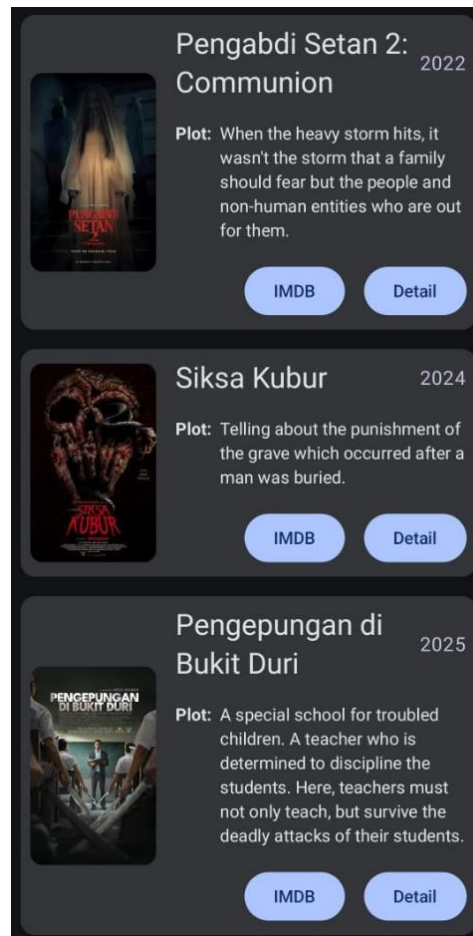
MODUL 3 : BUILD A SCROLLABLE LIST

SOAL 1

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

1. List menggunakan fungsi RecyclerView (XML) dan 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 untuk membuka laman detail item
5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
7. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)
8. Aplikasi berbasis XML harus menggunakan ViewBinding

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



Gambar 11. Contoh UI List

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



Gambar 12. Contoh UI Detail

A. Source Code

1) Versi Jetpack Compose

MainActivity.kt

Tabel 7. Source Code Jetpack Compose MainActivity.kt Soal 1

	<pre>package com.example.scrollablecompose import android.os.Bundle import androidx.activity.ComponentActivity import androidx.activity.compose.setContent import androidx.activity.enableEdgeToEdge import androidx.compose.runtime.Composable import androidx.navigation.compose.NavHost</pre>
--	---

```

import androidx.navigation.compose.composable
import
androidx.navigation.compose.rememberNavController
import
com.example.scrollablecompose.screens.DetailScreen
import
com.example.scrollablecompose.screens.ListScreen
import
com.example.scrollablecompose.ui.theme.ScrollableCom
poseTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            ScrollableComposeTheme {
                MyFavKamenRiderApp()
            }
        }
    }
}

@Composable
fun MyFavKamenRiderApp() {
    val navController = rememberNavController()
    NavHost(navController = navController,
startDestination = Routes.listScreen, builder = {
        composable(Routes.listScreen) {

```

	<pre> ListScreen(navController) } composable(Routes.detailScreen+"/{id}") { val idString = it.arguments?.getString("id") val id = idString?.toIntOrNull() ?: 0 DetailScreen(navController, id) } }) } </pre>
--	--

Card.kt

Tabel 8. Source Code Jetpack Compose Card.kt Soal 1

	<pre> package com.example.scrollablecompose.screens import android.content.Context import android.content.Intent import android.content.res.Configuration import android.net.Uri import androidx.compose.foundation.Image import androidx.compose.foundation.layout.Arrangement import androidx.compose.foundation.layout.Column import androidx.compose.foundation.layout.PaddingValues import androidx.compose.foundation.layout.Row import androidx.compose.foundation.layout.Spacer import androidx.compose.foundation.layout.fillMaxWidth </pre>
--	--

```

import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import
androidx.compose.foundation.layout.wrapContentHeight
import
androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import com.example.scrollablecompose.R
import com.example.scrollablecompose.Routes
import
com.example.scrollablecompose.model.KamenRider

```

```

@Composable
fun RiderCard(rider: KamenRider, navController:
NavController, orientationMode: Int) {
    val context = LocalContext.current

    if(orientationMode ==
Configuration.ORIENTATION_PORTRAIT) {
        RiderCardPortrait(rider, navController,
context)
    }
    else{
        RiderCardLandscape(rider, navController,
context)
    }
}

@Composable
fun RiderCardPortrait(rider: KamenRider,
navController: NavController, context: Context){
    Card(
        modifier = Modifier
            .padding(7.dp)
            .fillMaxWidth()
            .wrapContentHeight(),
        shape = MaterialTheme.shapes.medium,
        colors = CardDefaults.cardColors(
            containerColor =
MaterialTheme.colorScheme.surface

```

```

    ),
    elevation = CardDefaults.cardElevation(
        defaultElevation = 5.dp
    )
) {
    Row(
        verticalAlignment = Alignment.Top,
        modifier = Modifier.padding(5.dp)
    ) {
        Image(
            painter = painterResource(id =
rider.posterRes),
            contentDescription = "Poster
${rider.name}",
            modifier = Modifier
                .size(width = 120.dp, height =
150.dp)
                .padding(8.dp)

            .align(Alignment.CenterVertically)

            .clip(RoundedCornerShape(15.dp)),
            contentScale =
ContentScale.FillBounds,
        )
        Column(Modifier.padding(8.dp)) {
            Row (
                modifier = Modifier
                    .fillMaxWidth()
            ) {

```



```

        Text(
            text = rider.name,
            fontSize = 20.sp,
            fontWeight =
FontWeight.Bold,
            color =
MaterialTheme.colorScheme.onSurface,
            modifier = Modifier
                .align(Alignment.CenterVertically)
                .weight(1f)
        )
        Spacer(Modifier.width(20.dp))
        Text(
            text =
rider.year.toString(),
            style =
MaterialTheme.typography.labelMedium,
            fontSize = 15.sp,
            color =
MaterialTheme.colorScheme.onSurface,
            modifier =
Modifier.align(Alignment.CenterVertically)
        )
    }

    Spacer(Modifier.height(8.dp))
    Text(
        text =
stringResource(R.string.description),

```

	style	=
	MaterialTheme.typography.labelMedium,	
	color	=
	MaterialTheme.colorScheme.onSurface	
)	
	Text (
	text = rider.description,	
	style	=
	MaterialTheme.typography.bodySmall,	
	color	=
	MaterialTheme.colorScheme.onSurface,	
	textAlign = TextAlign.Justify	
)	
	Spacer (Modifier.height (8.dp))	
	Row (
	Modifier.fillMaxWidth(),	
	horizontalArrangement	=
	Arrangement.End	
) {	
	Button (
	shape	=
	RoundedCornerShape (12.dp),	
	contentPadding	=
	PaddingValues (horizontal = 15.dp),	
	onClick = {	
	val intent	=
	Intent (Intent.ACTION_VIEW)	
	intent.data	=
	Uri.parse (rider.imdbUrl)	

```
intent.setPackage("com.android.chrome")

context.startActivity(intent)
    }
) {

Text(stringResource(R.string.imdb),      fontSize      =
13.sp)

    }
    Spacer(Modifier.width(10.dp))
    Button(
        shape      =
RoundedCornerShape(12.dp),
        contentPadding      =
PaddingValues(horizontal = 15.dp),
        onClick = {

navController.navigate(Routes.detailScreen+"/${rider
.id}")

    }
) {

Text(stringResource(R.string.detail),      fontSize      =
13.sp)

    }

}

}

}
```

```

}

@Composable
fun RiderCardLandscape(rider: KamenRider,
navController: NavController, context: Context){
    Card(
        modifier = Modifier
            .padding(7.dp)
            .fillMaxWidth()
            .wrapContentHeight(),
        shape = MaterialTheme.shapes.medium,
        colors = CardDefaults.cardColors(
            containerColor =
MaterialTheme.colorScheme.surface
        ),
        elevation = CardDefaults.cardElevation(
            defaultElevation = 5.dp
        )
    ) {
        Row(
            verticalAlignment = Alignment.Top,
            modifier = Modifier.padding(5.dp)
        ) {
            Image(
                painter = painterResource(id =
rider.posterRes),
                contentDescription = "Poster
${rider.name}",
                modifier = Modifier

```

```

                .size(width = 192.dp, height =
240.dp)

                .padding(8.dp)

                .align(Alignment.CenterVertically)

                .clip(RoundedCornerShape(15.dp)),
                    contentScale
ContentScale.FillBounds,
                )
                Spacer(Modifier.width(5.dp))
                Column(Modifier.padding(8.dp)) {
                    Row (
                        modifier = Modifier
                            .fillMaxWidth()
                    ) {
                        Text(
                            text = rider.name,
                            fontSize = 30.sp,
                            fontWeight
FontWeight.Bold,
                            color
MaterialTheme.colorScheme.onSurface,
                            modifier = Modifier

                        .align(Alignment.CenterVertically)
                            .weight(1f)
                    )
                    Spacer(Modifier.width(20.dp))
                    Text (

```

```

        text
        =
rider.year.toString(),
        style
        =
MaterialTheme.typography.labelMedium,
        fontSize = 20.sp,
        color
        =
MaterialTheme.colorScheme.onSurface,
        modifier
        =
Modifier.align(Alignment.CenterVertically)
    )
}

    Spacer(Modifier.height(15.dp))
    Text(
        text
        =
stringResource(R.string.description),
        style
        =
MaterialTheme.typography.labelMedium,
        fontSize = 20.sp,
        color
        =
MaterialTheme.colorScheme.onSurface
    )
    Spacer(Modifier.height(5.dp))
    Text(
        text = rider.description,
        style
        =
MaterialTheme.typography.bodyMedium,
        fontSize = 18.sp,
        color
        =
MaterialTheme.colorScheme.onSurface,

```

	<pre> textAlign = TextAlign.Justify) Spacer (Modifier.height (50.dp)) Row (Modifier.fillMaxWidth(), horizontalArrangement Arrangement.End) { Button(shape RoundedCornerShape (16.dp), contentPadding PaddingValues (vertical = 15.dp, horizontal = 20.dp), onClick = { val intent Intent (Intent.ACTION_VIEW) intent.data Uri.parse (rider.imdbUrl) intent.setPackage ("com.android.chrome") context.startActivity(intent) }) { Text (stringResource (R.string.imdb), fontSize 18.sp) } Spacer (Modifier.width (10.dp)) </pre>
--	---

	<pre> Button(shape = RoundedCornerShape(16.dp), contentPadding = PaddingValues(vertical = 15.dp, horizontal = 20.dp), onClick = { navController.navigate(Routes.detailScreen+"/\${rider .id}") }) { Text(stringResource(R.string.detail), fontSize = 18.sp) } } } } } } } } </pre>
--	--

ListScreen.kt

Tabel 9. Source Code Jetpack Compose ListScreen.kt Soal 1

	<pre> package com.example.scrollablecompose.screens import androidx.compose.foundation.layout.PaddingValues import androidx.compose.foundation.layout.fillMaxWidth </pre>
--	--


```

import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import
androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableIntStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import
androidx.compose.ui.platform.LocalConfiguration
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import
androidx.navigation.compose.rememberNavController
import com.example.scrollablecompose.R
import
com.example.scrollablecompose.data.getKamenRiderList
import
com.example.scrollablecompose.ui.theme.ScrollableCompo
seTheme

```

```

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ListScreen(navController: NavController) {
    val config = LocalConfiguration.current
    val orientationMode by remember {
mutableIntStateOf(config.orientation) }

    Scaffold(
        topBar = {
            TopAppBar(
                title = {
Text(stringResource(R.string.topappbarr_title))
                },
                colors =
TopAppBarDefaults.topAppBarColors(
                    containerColor =
MaterialTheme.colorScheme.primary,
                    titleContentColor =
MaterialTheme.colorScheme.onPrimary
                )
            )
        }
    ) { innerPadding ->
        LazyColumn(
            modifier = Modifier
                .fillMaxWidth()
                .padding(innerPadding),
            contentPadding = PaddingValues(16.dp)
        ) {

```

```

        items(getKamenRiderList()) { rider ->
            RiderCard(rider, navController,
orientationMode)
        }
    }
}

@Preview(
    showBackground = true,
    name = "Redmi Note 13",
    widthDp = 390,
    heightDp = 800
)
@Composable
fun ListScreenPortraitPreview() {
    ScrollableComposeTheme {
        val navController = rememberNavController()
        ListScreen(navController = navController)
    }
}

@Preview(
    showBackground = true,
    name = "Redmi Note 13",
    widthDp = 800,
    heightDp = 390
)
@Composable
fun ListScreenLandscapePreview() {

```

	<pre> ScrollableComposeTheme { val navController = rememberNavController() ListScreen(navController = navController) } </pre>
--	---

DetailScreen.kt

Tabel 10. Source Code Jetpack Compose Detail.kt Soal 1

	<pre> package com.example.scrollablecompose.screens import android.content.res.Configuration import androidx.compose.foundation.Image import androidx.compose.foundation.layout.Column import androidx.compose.foundation.layout.PaddingValues import androidx.compose.foundation.layout.Row import androidx.compose.foundation.layout.Spacer import androidx.compose.foundation.layout.fillMaxSize import androidx.compose.foundation.layout.fillMaxWidth import androidx.compose.foundation.layout.height import androidx.compose.foundation.layout.padding import androidx.compose.foundation.layout.size import androidx.compose.foundation.layout.width import androidx.compose.foundation.rememberScrollState import androidx.compose.foundation.shape.RoundedCornerShape import androidx.compose.foundation.verticalScroll </pre>
--	---

```

import androidx.compose.material.icons.Icons
import
androidx.compose.material.icons.automirrored.filled.ArrowBack
import
androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TopAppBar
import androidx.compose.material3.TopAppBarDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableIntStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import
androidx.compose.ui.platform.LocalConfiguration
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

```

```

import androidx.navigation.NavController
import
androidx.navigation.compose.rememberNavController
import com.example.scrollablecompose.R
import
com.example.scrollablecompose.data.getKamenRiderList
import com.example.scrollablecompose.model.KamenRider
import
com.example.scrollablecompose.ui.theme.ScrollableCom
poseTheme

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DetailScreen(navController: NavController, id:
Int) {
    val rider = getKamenRiderList().find { it.id == id
    }

    val config = LocalConfiguration.current
    val orientationMode by remember {
mutableIntStateOf(config.orientation) }

    Scaffold (
        topBar = {
            TopAppBar(
                title = {

Text(stringResource(R.string.detail))

                },
                navigationIcon = {

```

```

        IconButton(      onClick      =      {
navController.navigateUp() } ) {
            Icon(
                imageVector      =
Icons.AutoMirrored.Filled.ArrowBack,
                contentDescription = null
            )
        },
        colors      =
TopAppBarDefaults.topAppBarColors(
            containerColor      =
MaterialTheme.colorScheme.primary,
            titleContentColor      =
MaterialTheme.colorScheme.onPrimary,
            navigationIconContentColor      =
MaterialTheme.colorScheme.onPrimary
        )
    )
}
) { padding ->
    if (rider != null) {
        if(orientationMode      ==
Configuration.ORIENTATION_PORTRAIT) {
            DetailPortraitLayout(rider, padding)
        }
        else{
            DetailLandscapeLayout(rider, padding)
        }
    }
}

```

```

        else{
            Text(
                text = "Data not found",
                fontSize = 25.sp
            )
        }
    }
}

@Composable
fun DetailPortraitLayout(rider : KamenRider,
paddingValues: PaddingValues) {
    val scrollState = rememberScrollState()

    Column (
        modifier = Modifier
            .padding(paddingValues)
            .padding(vertical = 15.dp, horizontal =
20.dp)
            .verticalScroll(scrollState)
            .fillMaxSize(),
        horizontalAlignment =
Alignment.CenterHorizontally
    ) {
        Image(
            painter =
painterResource(rider.posterRes),
            contentDescription = "Poster
${rider.name}",
            contentScale = ContentScale.FillBounds,

```



```

        modifier = Modifier
            .size(width = 300.dp, height = 400.dp)
            .clip(RoundedCornerShape(25.dp))
    )
    Spacer(Modifier.height(15.dp))
    Text(
        text = rider.name,
        fontSize = 30.sp,
        fontWeight = FontWeight.Bold,
        textAlign = TextAlign.Center
    )
    Spacer(Modifier.height(20.dp))
    Column (
        modifier = Modifier.fillMaxWidth()
    ) {
        Text(
            text = stringResource(R.string.year,
rider.year),

            fontSize = 18.sp,
            fontWeight = FontWeight.Bold
        )

        Spacer(Modifier.height(15.dp))

        Text(
            text
=
stringResource(R.string.description),
            fontSize = 18.sp,
            fontWeight = FontWeight.Bold
        )
    }

```

```

        Spacer(Modifier.height(3.dp))
        Text(
            text = rider.description,
            textAlign = TextAlign.Justify
        )
    }
}

@Composable
fun DetailLandscapeLayout(rider: KamenRider,
paddingValues: PaddingValues) {
    val scrollState = rememberScrollState()

    Row (
        modifier = Modifier
            .fillMaxWidth()
            .verticalScroll(scrollState)
            .padding(paddingValues)
            .padding(vertical = 15.dp, horizontal =
20.dp)
    ) {
        Image(
            painter = painterResource(id =
rider.posterRes),
            contentDescription = "Poster
${rider.name}",
            modifier = Modifier
                .size(width = 240.dp, height = 300.dp)
                .padding(8.dp)
        )
    }
}

```

```

        .align(Alignment.CenterVertically)
        .clip(RoundedCornerShape(15.dp)),
        contentScale = ContentScale.FillBounds,
    )
    Spacer(Modifier.width(5.dp))
    Column(Modifier.padding(8.dp)) {

        Text(
            text = rider.name,
            fontSize = 35.sp,
            fontWeight = FontWeight.Bold,
        )

        Spacer(Modifier.height(15.dp))
        Text(
            text = stringResource(R.string.year,
rider.year),
            fontSize = 21.sp,
            fontWeight = FontWeight.Bold
        )

        Spacer(Modifier.height(15.dp))
        Text(
            text
=
stringResource(R.string.description),
            fontSize = 21.sp,
            fontWeight = FontWeight.Bold,
            color
=
MaterialTheme.colorScheme.onSurface
        )
    }

```

```

        Text(
            text = rider.description,
            fontSize = 20.sp,
            color =
MaterialTheme.colorScheme.onSurface,
            textAlign = TextAlign.Justify
        )
    }
}

}

@Preview(
    showBackground = true,
    widthDp = 390,
    heightDp = 800,
    name = "Redmi Note 13 Portrait"
)
@Composable
fun DetailScreenPortraitPreview() {
    ScrollableComposeTheme {
        val navController = rememberNavController()
        DetailScreen(navController,
getKamenRiderList()[0].id)
    }
}

@Preview(
    showBackground = true,
    widthDp = 800,

```

```

        heightDp = 390,
        name = "Redmi Note 13 Landscape"
    )
    @Composable
    fun DetailScreenLandscapePreview() {
        ScrollableComposeTheme {
            val navController = rememberNavController()
            DetailScreen(navController,
getKamenRiderList()[0].id)
        }
    }

```

Routes.kt

Tabel 11. Source Code Jetpack Compose Routes.kt Soal 1

```

package com.example.scrollablecompose

object Routes {
    val listScreen = "list_screen"
    val detailScreen = "detail_screen"
}

```

KamenRider.kt

Tabel 12. Source Code Jetpack Compose KamenRider.kt Soal 1

```

package com.example.scrollablecompose.model

data class KamenRider (
    val id: Int,
    val name: String,
    val description: String,

```

	<pre> val year: Int, val imdbUrl: String, val posterRes : Int) </pre>
--	--

KamenRiderList.kt

Tabel 13. Source Code Jetpack Compose KamenRiderList.kt Soal 1

	<pre> package com.example.scrollablecompose.data import com.example.scrollablecompose.R import com.example.scrollablecompose.model.KamenRider fun getKamenRiderList(): List<KamenRider> { return listOf(KamenRider(id = 1, name = "Kamen Rider Geats", description = "Para peserta bertarung dalam Desire Grand Prix untuk menjadi pahlawan ideal di dunia seperti permainan.", year = 2022, imdbUrl = "https://www.imdb.com/title/tt20758104/?ref_=ext_shr_ lnk", posterRes = R.drawable.kr_geats), KamenRider(id = 2, name = "Kamen Rider Build", </pre>
--	--

	<pre> description = "Seorang fisikawan jenius berubah menjadi Build untuk mengungkap kebenaran di balik Kotak Pandora.", year = 2017, imdbUrl = "https://www.imdb.com/title/tt6982472/?ref_ext_shr_l nk", posterRes = R.drawable.kr_build), KamenRider(id = 3, name = "Kamen Rider Zero One", description = "Di dunia yang dipenuhi AI dan robot, Aruto bertarung sebagai Zero-One demi melindungi manusia dan Humagear.", year = 2019, imdbUrl = "https://www.imdb.com/title/tt10333650/?ref_ext_shr_ lnk", posterRes = R.drawable.kr_zero_one), KamenRider(id = 4, name = "Kamen Rider W (Double)", description = "Dua jiwa dalam satu tubuh, Shotaro dan Philip bekerja sama sebagai detektif untuk melindungi Futo dari kejahatan Gaia Memory.", year = 2009, </pre>
--	--

	<pre> imdbUrl = "https://www.imdb.com/title/tt1483620/?ref_ext_shrink", posterRes = R.drawable.kr_double), KamenRider(id = 5, name = "Kamen Rider Gaim", description = "Para Rider bertarung dalam permainan yang tampak seperti hiburan, namun berubah menjadi konflik serius demi menyelamatkan dunia.", year = 2013, imdbUrl = "https://www.imdb.com/title/tt3079058/?ref_ext_shrink", posterRes = R.drawable.kr_gaim), KamenRider(id = 6, name = "Kamen Rider Ex-Aid", description = "Seorang dokter sekaligus gamer bertarung melawan virus digital Bugster demi menyelamatkan pasien dari penyakit misterius.", year = 2016, imdbUrl = "https://www.imdb.com/title/tt5813014/?ref_ext_shrink", posterRes = R.drawable.kr_ex_aid), KamenRider(</pre>
--	--

	<pre> id = 7, name = "Kamen Rider Gotchard", year = 2023, description = "Rinne dan Houtarou bertarung bersama menggunakan kartu Chemies untuk menjaga keseimbangan dunia.", imdbUrl = "https://www.imdb.com/title/tt27830205/?ref_=ext_shr_ lnk", posterRes = R.drawable.kr_gotchard) } </pre>
--	---

2) Versi XML

MainActivity.kt

Tabel 14. Source Code XML MainActivity.kt

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

12	}
13	}

MainAdapter.kt

Tabel 15. Source Code XML MainAdapter.kt

1	package com.example.scrollablexml
2	
3	import android.view.LayoutInflater
4	import android.view.View
5	import android.view.ViewGroup
6	import android.widget.Button
7	import android.widget.ImageView
8	import android.widget.TextView
9	import androidx.recyclerview.widget.RecyclerView
10	import com.example.scrollablexml.model.KamenRider
11	
12	class MainAdapter (
13	private val listRider: List<KamenRider>,
14	private val onImdbClick: (String) -> Unit,
15	private val onDetailClick: (Int) -> Unit
16	
17) : RecyclerView.Adapter<MainAdapter.ViewHolder>() {
18	class ViewHolder(view: View):
19	RecyclerView.ViewHolder(view) {
20	val riderImage =
21	view.findViewById<ImageView>(R.id.riderImage)
22	val riderName =
23	view.findViewById<TextView>(R.id.riderName)
24	val riderYear =
25	view.findViewById<TextView>(R.id.riderYear)

26	val	riderDesc	=
27	view.findViewById<TextView>(R.id.descBody)		
28	val	imdbBtn	=
29	view.findViewById<Button>(R.id.imdbBtn)		
30	val	detailBtn	=
31	view.findViewById<Button>(R.id.detailBtn)		
32	}		
33			
34	override fun	onCreateViewHolder(parent:	
35	ViewGroup, viewType: Int): ViewHolder {		
36	return		
37	ViewHolder(LayoutInflater.from(parent.context).inflat		
38	ate(R.layout.adapter_main, parent, false))		
	}		
	override fun getItemCount() = listRider.size		
	override fun	onBindViewHolder(holder:	
	ViewHolder, position: Int) {		
	val rider = listRider[position]		
	holder.riderImage.setImageResource(rider.posterRes)		
	holder.riderName.text = rider.name		
	holder.riderYear.text = "\${rider.year}"		
	holder.riderDesc.text = rider.description		
	holder.imdbBtn.setOnClickListener	{	
	onImdbClick(rider.imdbUrl) }		
	holder.detailBtn.setOnClickListener	{	
	onDetailClick(rider.id) }		
	}		

	}
--	---

ListFragment.kt

Tabel 16. Source Code XML ListFragment.kt

1	package com.example.scrollablexml
2	
3	import android.content.Intent
4	import android.net.Uri
5	import android.os.Bundle
6	import androidx.fragment.app.Fragment
7	import android.view.LayoutInflater
8	import android.view.View
9	import android.view.ViewGroup
10	import
11	androidx.navigation.fragment.findNavController
12	import androidx.recyclerview.widget.RecyclerView
13	import
14	com.example.scrollablexml.data.getKamenRiderList
15	
16	// TODO: Rename parameter arguments, choose names
17	that match
18	// the fragment initialization parameters, e.g.
19	ARG_ITEM_NUMBER
20	private const val ARG_PARAM1 = "param1"
21	private const val ARG_PARAM2 = "param2"
22	
23	/**
24	* A simple [Fragment] subclass.
25	* Use the [ListFragment.newInstance] factory method
26	to

```

27  * create an instance of this fragment.
28  */
29  class ListFragment : Fragment() {
30      // TODO: Rename and change types of parameters
31      private var param1: String? = null
32      private var param2: String? = null
33
34      override fun onCreate(savedInstanceState:
35 Bundle?) {
36          super.onCreate(savedInstanceState)
37          arguments?.let {
38              param1 = it.getString(ARG_PARAM1)
39              param2 = it.getString(ARG_PARAM2)
40          }
41      }
42
43      override fun onCreateView(
44          inflater: LayoutInflater, container:
45 ViewGroup?,
46          savedInstanceState: Bundle?
47      ): View? {
48          // Inflate the layout for this fragment
49          val view =
50 inflater.inflate(R.layout.fragment_list, container,
51 false)
52          val recyclerView =
53 view.findViewById<RecyclerView>(R.id.recyclerView)
54
55          val mainAdapter = MainAdapter(
56              listRider = getKamenRiderList(),

```

57	onDetailClick = { riderId ->
58	val action =
59	ListFragmentDirections.actionListFragmentToDetailFr
60	agment(riderId)
61	
62	findNavController().navigate(action)
63	},
64	onImdbClick = { imdbUrl ->
65	val intent =
66	Intent(Intent.ACTION_VIEW, Uri.parse(imdbUrl)).apply
67	{
68	
69	setPackage("com.android.chrome")
70	}
71	startActivity(intent)
72	}
73)
74	
75	recyclerView.adapter = mainAdapter
76	return view
77	}
78	
79	
80	companion object {
81	/**
82	* Use this factory method to create a new
	instance of
	* this fragment using the provided
	parameters.
	*

	<pre> * @param param1 Parameter 1. * @param param2 Parameter 2. * @return A new instance of fragment listFragment. */ // TODO: Rename and change types and number of parameters @JvmStatic fun newInstance(param1: String, param2: String) = ListFragment().apply { arguments = Bundle().apply { putString(ARG_PARAM1, param1) putString(ARG_PARAM2, param2) } } } } </pre>
--	---

DetailFragment.kt

Tabel 17. Source Code XML DetailFragment.kt

1	package com.example.scrollablexml
2	
3	import android.os.Bundle
4	import androidx.fragment.app.Fragment
5	import android.view.LayoutInflater
6	import android.view.View
7	import android.view.ViewGroup
8	import android.widget.TextView
9	import androidx.navigation.fragment.navArgs

```

10 import
11 com.example.scrollablexml.data.getKamenRiderList
12 import
13 com.google.android.material.appbar.MaterialToolbar
14 import
15 com.google.android.material.imageview.ShapeableImage
16 view
17
18 // TODO: Rename parameter arguments, choose names
19 that match
20 // the fragment initialization parameters, e.g.
21 ARG_ITEM_NUMBER
22 private const val ARG_PARAM1 = "param1"
23 private const val ARG_PARAM2 = "param2"
24
25 /**
26  * A simple [Fragment] subclass.
27  * Use the [DetailFragment.newInstance] factory
28 method to
29  * create an instance of this fragment.
30  */
31 class DetailFragment : Fragment() {
32     // TODO: Rename and change types of parameters
33     private var param1: String? = null
34     private var param2: String? = null
35     private val args: DetailFragmentArgs by navArgs()
36
37     override fun onCreate(savedInstanceState:
38 Bundle?) {
39         super.onCreate(savedInstanceState)

```


40	arguments?.let {
41	param1 = it.getString(ARG_PARAM1)
42	param2 = it.getString(ARG_PARAM2)
43	}
44	}
45	
46	override fun onCreateView(
47	inflater: LayoutInflater, container:
	ViewGroup?,
	savedInstanceState: Bundle?
): View? {
	// Inflate the layout for this fragment
	val view =
	inflater.inflate(R.layout.fragment_detail,
	container, false)
	val toolbar =
	view.findViewById<MaterialToolbar>(R.id.topAppBar)
	toolbar.setNavigationOnClickListener {
	requireActivity().onBackPressedDispatcher.onBackPressed()
	}
	val rider = getKamenRiderList().find { it.id
	== args.riderId }
	view.findViewById<ShapeableImageView>(R.id.riderImage).setImageResource(rider?.posterRes ?: 0)

```

view.findViewById<TextView>(R.id.riderName).text    =
rider?.name

view.findViewById<TextView>(R.id.riderYear).text    =
getString(R.string.detail_year, rider?.year)

view.findViewById<TextView>(R.id.descBody).text     =
rider?.description

        return view
    }

    companion object {
        /**
         * Use this factory method to create a new
instance of
         * this fragment using the provided
parameters.
         *
         * @param param1 Parameter 1.
         * @param param2 Parameter 2.
         * @return A new instance of fragment
detailFragment.
         */
        // TODO: Rename and change types and number
of parameters
        @JvmStatic
        fun newInstance(param1: String, param2:
String) =

```

	<pre> DetailFragment().apply { arguments = Bundle().apply { putString(ARG_PARAM1, param1) putString(ARG_PARAM2, param2) } } } } </pre>
--	--

KamenRider.kt

Tabel 18. Source Code XML KamenRider.kt

1	package com.example.scrollablexml.model
2	
3	data class KamenRider(
4	val id: Int,
5	val name: String,
6	val description: String,
7	val year: Int,
8	val imdbUrl: String,
9	val posterRes : Int
10)

KamenRiderList.kt

Tabel 19. Source Code XML KamenRiderList.kt

1	package com.example.scrollablexml.data
2	
3	import com.example.scrollablexml.R
4	import com.example.scrollablexml.model.KamenRider
5	

6	fun getKamenRiderList(): List<KamenRider> {
7	return listOf(
8	KamenRider(
9	id = 1,
10	name = "Kamen Rider Geats",
11	description = "Para peserta bertarung
12	dalam Desire Grand Prix untuk menjadi pahlawan ideal
13	di dunia seperti permainan.",
14	year = 2022,
15	imdbUrl =
16	"https://www.imdb.com/title/tt20758104/?ref_ext_s
17	hr_lnk",
18	posterRes = R.drawable.kr_geats
19),
20	KamenRider(
21	id = 2,
22	name = "Kamen Rider Build",
23	description = "Seorang fisikawan jenius
24	berubah menjadi Build untuk mengungkap kebenaran di
25	balik Kotak Pandora.",
26	year = 2017,
27	imdbUrl =
28	"https://www.imdb.com/title/tt6982472/?ref_ext_sh
29	r_lnk",
30	posterRes = R.drawable.kr_build
31),
32	KamenRider(
33	id = 3,
34	name = "Kamen Rider Zero One",
35	

36	description = "Di dunia yang dipenuhi AI
37	dan robot, Aruto bertarung sebagai Zero-One demi
38	melindungi manusia dan Humagear.",
39	year = 2019,
40	imdbUrl =
41	"https://www.imdb.com/title/tt10333650/?ref_ext_s
42	hr_lnk",
43	posterRes = R.drawable.kr_zero_one
44),
45	KamenRider(
46	id = 4,
47	name = "Kamen Rider W (Double)",
48	description = "Dua jiwa dalam satu
49	tubuh, Shotaro dan Philip bekerja sama sebagai
50	detektif untuk melindungi Futo dari kejahatan Gaia
51	Memory.",
52	year = 2009,
53	imdbUrl =
54	"https://www.imdb.com/title/tt1483620/?ref_ext_sh
55	r_lnk",
56	posterRes = R.drawable.kr_double
57),
58	KamenRider(
59	id = 5,
60	name = "Kamen Rider Gaim",
61	description = "Para Rider bertarung
62	dalam permainan yang tampak seperti hiburan, namun
63	berubah menjadi konflik serius demi menyelamatkan
64	dunia.",
65	year = 2013,

	<pre> imdbUrl = "https://www.imdb.com/title/tt3079058/?ref_=ext_sh r_lnk", posterRes = R.drawable.kr_gaim), KamenRider(id = 6, name = "Kamen Rider Ex-Aid", description = "Seorang dokter sekaligus gamer bertarung melawan virus digital Bugster demi menyelamatkan pasien dari penyakit misterius.", year = 2016, imdbUrl = "https://www.imdb.com/title/tt5813014/?ref_=ext_sh r_lnk", posterRes = R.drawable.kr_ex_aid), KamenRider(id = 7, name = "Kamen Rider Gotchard", year = 2023, description = "Rinne dan Houtarou bertarung bersama menggunakan kartu Chemies untuk menjaga keseimbangan dunia.", imdbUrl = "https://www.imdb.com/title/tt27830205/?ref_=ext_s hr_lnk", posterRes = R.drawable.kr_gotchard)) </pre>
--	--

	}
--	---

activity_main.xml

Tabel 20. Source Code XML activity_main.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	
4	xmlns:android="http://schemas.android.com/apk/res/
5	android"
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:fitsSystemWindows="true"
11	android:layout_width="match_parent"
12	android:layout_height="match_parent"
13	tools:context=".MainActivity">
14	
15	<androidx.fragment.app.FragmentContainerView
16	android:id="@+id/fragmentContainerView6"
17	
18	android:name="androidx.navigation.fragment.NavHost
19	Fragment"
20	android:layout_width="0dp"
21	android:layout_height="0dp"
22	app:defaultNavHost="true"
23	app:navGraph="@navigation/main_graph"
24	app:layout_constraintTop_toTopOf="parent"
	app:layout_constraintBottom_toBottomOf="parent"

	<pre> app:layout_constraintStart_toStartOf="parent" app:layout_constraintEnd_toEndOf="parent"/> </androidx.constraintlayout.widget.ConstraintLayout> </pre>
--	---

adapter_main.xml

Tabel 21. Source Code XML adapter_main.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	xmlns:android="http://schemas.android.com/apk/res/
4	android"
5	android:layout_width="match_parent"
6	android:layout_height="wrap_content"
7	xmlns:app="http://schemas.android.com/apk/res-
8	auto"
9	
10	xmlns:tools="http://schemas.android.com/tools">
11	
12	<androidx.cardview.widget.CardView
13	android:id="@+id/cardView"
14	android:layout_width="match_parent"
15	android:layout_height="wrap_content"
16	app:layout_constraintTop_toTopOf="parent"
17	
18	app:layout_constraintBottom_toBottomOf="parent"
19	
20	app:layout_constraintStart_toStartOf="parent"
21	app:layout_constraintEnd_toEndOf="parent"

22	android:layout_margin="16dp"
23	
24	app:cardBackgroundColor="?android:attr/colorBackgr
25	ound"
26	app:cardElevation="4dp"
27	app:cardCornerRadius="15dp">
28	
29	
30	<androidx.constraintlayout.widget.ConstraintLayout
31	android:layout_width="match_parent"
32	android:layout_height="match_parent"
33	android:padding="15dp">
34	
35	
36	<com.google.android.material.imageview.ShapeableIm
37	ageView
38	android:id="@+id/riderImage"
39	android:layout_width="120dp"
40	android:layout_height="150dp"
41	android:scaleType="centerCrop"
42	android:src="@drawable/kr_geats"
43	
44	app:layout_constraintBottom_toBottomOf="parent"
45	
46	app:layout_constraintStart_toStartOf="parent"
47	
48	app:layout_constraintTop_toTopOf="parent"
49	
50	app:shapeAppearanceOverlay="@style/RoundedImageVie
51	w"/>

52	
53	
54	
55	<androidx.constraintlayout.widget.ConstraintLayout
56	android:layout_width="0dp"
57	
58	android:layout_height="match_parent"
59	
60	app:layout_constraintTop_toTopOf="parent"
61	
62	app:layout_constraintEnd_toEndOf="parent"
63	
64	app:layout_constraintStart_toEndOf="@id/riderImage
65	"
66	android:layout_marginStart="15dp">
67	
68	
69	<androidx.constraintlayout.widget.ConstraintLayout
70	
71	android:id="@+id/titleYearLayout"
72	android:layout_width="0dp"
73	
74	android:layout_height="wrap_content"
75	
76	app:layout_constraintStart_toStartOf="parent"
77	
78	app:layout_constraintEnd_toEndOf="parent"
79	
80	app:layout_constraintTop_toTopOf="parent">
81	

82	<TextView
83	
84	android:id="@+id/riderName"
85	android:layout_width="0dp"
86	
87	android:layout_height="wrap_content"
88	
89	app:layout_constraintTop_toTopOf="parent"
90	
91	app:layout_constraintBottom_toBottomOf="parent"
92	
93	app:layout_constraintStart_toStartOf="parent"
94	
95	app:layout_constraintEnd_toStartOf="@id/riderYear"
96	
97	app:layout_constraintHorizontal_weight="6"
98	android:textSize="20sp"
99	tools:text="Kamen Rider
100	Geats"
101	android:textStyle="bold" />
102	
103	<TextView
104	
105	android:id="@+id/riderYear"
106	android:layout_width="0dp"
107	
108	android:layout_height="wrap_content"
109	
110	app:layout_constraintTop_toTopOf="parent"
111	

112	
113	app:layout_constraintBottom_toBottomOf="parent"
114	
115	app:layout_constraintStart_toEndOf="@id/riderName"
116	
117	app:layout_constraintEnd_toEndOf="parent"
118	
119	app:layout_constraintHorizontal_weight="2"
120	android:gravity="end"
121	android:textSize="16sp"
122	android:textStyle="bold"
123	tools:text="2022" />
124	
125	
126	</androidx.constraintlayout.widget.ConstraintLayout>
127	
128	
129	<TextView
130	android:id="@+id/descTitle"
131	
132	android:layout_width="wrap_content"
133	
134	android:layout_height="wrap_content"
135	
136	app:layout_constraintTop_toBottomOf="@id/titleYear
137	Layout"
138	
139	app:layout_constraintStart_toStartOf="parent"
140	android:layout_marginTop="8dp"
141	android:text="@string/desc"

142	android:textStyle="bold" />
143	
144	<TextView
145	android:id="@+id/descBody"
	android:layout_width="0dp"
	android:layout_height="wrap_content"
	app:layout_constraintTop_toBottomOf="@id/descTitle"
	"
	app:layout_constraintStart_toStartOf="parent"
	app:layout_constraintEnd_toEndOf="parent"
	android:layout_marginTop="3dp"
	android:justificationMode="inter_word"
	tools:text="Deskripsi panjang
	lorem ipsum lorem ipsum lorem ipsum lorem ipsum
	lorem ipsum" />
	<androidx.constraintlayout.widget.ConstraintLayout
	android:layout_width="0dp"
	android:layout_height="wrap_content"
	app:layout_constraintTop_toBottomOf="@id/descBody"
	app:layout_constraintStart_toStartOf="parent"

```

app:layout_constraintEnd_toEndOf="parent"

android:layout_marginTop="15dp">

        <Button
            android:id="@+id/imdbBtn"

android:layout_width="80dp"

android:layout_height="wrap_content"

android:layout_marginEnd="5dp"
            android:textSize="12sp"
            app:cornerRadius="15dp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toStartOf="@id/detailBtn"

app:layout_constraintTop_toTopOf="parent"

android:text="@string/imdb" />

        <Button

android:id="@+id/detailBtn"

android:layout_width="80dp"

```

```
android:layout_height="wrap_content"
            android:textSize="12sp"
            app:cornerRadius="15dp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toTopOf="parent"

android:text="@string/detail" />

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

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

	<code></androidx.constraintlayout.widget.ConstraintLayout></code>
--	---

fragment_list.xml

Tabel 22. Source Code XML fragment_list.xml

1	<code><?xml version="1.0" encoding="utf-8"?></code>
2	<code><androidx.constraintlayout.widget.ConstraintLayout</code>
3	
4	<code>xmlns:android="http://schemas.android.com/apk/res/</code>
5	<code>android"</code>
6	<code>xmlns:app="http://schemas.android.com/apk/res-</code>
7	<code>auto"</code>
8	<code>xmlns:tools="http://schemas.android.com/tools"</code>
9	<code>android:layout_width="match_parent"</code>
10	<code>android:layout_height="match_parent"</code>
11	<code>tools:context=".ListFragment"></code>
12	
13	
14	<code><com.google.android.material.appbar.AppBarLayout</code>
15	<code>android:id="@+id/topAppBar"</code>
16	<code>android:layout_width="0dp"</code>
17	<code>android:layout_height="wrap_content"</code>
18	<code>android:background="?attr/colorPrimary"</code>
19	<code>app:layout_constraintTop_toTopOf="parent"</code>
20	
21	<code>app:layout_constraintStart_toStartOf="parent"</code>
22	<code>app:layout_constraintEnd_toEndOf="parent"></code>
23	
24	
25	

26	
27	<com.google.android.material.appbar.MaterialToolba
28	r
29	android:layout_width="match_parent"
30	android:layout_height="match_parent"
31	app:title="My Favorite Kamen Rider"
32	
33	app:titleTextColor="?attr/colorOnPrimary"/>
34	
35	</com.google.android.material.appbar.AppBarLayout>
36	
37	<androidx.recyclerview.widget.RecyclerView
38	android:id="@+id/recyclerView"
39	android:layout_width="0dp"
	android:layout_height="0dp"
	android:padding="10dp"
	tools:listitem="@layout/adapter_main"
	tools:itemCount="7"
	app:layoutManager="androidx.recyclerview.widget.Li
	nearLayoutManager"
	app:layout_constraintTop_toBottomOf="@id/topAppBar
	"
	app:layout_constraintBottom_toBottomOf="parent"
	app:layout_constraintStart_toStartOf="parent"
	app:layout_constraintEnd_toEndOf="parent"/>

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

fragment_detail.xml

Tabel 23. Source Code XML fragment_detail.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	
4	xmlns:android="http://schemas.android.com/apk/res/
5	android"
6	xmlns:app="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="match_parent"
11	tools:context=".DetailFragment">
12	
13	
14	<com.google.android.material.appbar.AppBarLayout
15	android:id="@+id/topAppBarLayout"
16	android:layout_width="0dp"
17	android:layout_height="wrap_content"
18	android:background="?attr/colorPrimary"
19	app:layout_constraintTop_toTopOf="parent"
20	
21	app:layout_constraintStart_toStartOf="parent"
22	app:layout_constraintEnd_toEndOf="parent">
23	
24	
25	

26	
27	<com.google.android.material.appbar.MaterialToolba
28	r
29	android:id="@+id/topAppBar"
30	android:layout_width="match_parent"
31	android:layout_height="match_parent"
32	app:title="Detail"
33	
34	app:titleTextColor="?attr/colorOnPrimary"
35	
36	app:navigationIcon="@drawable/baseline_arrow_back_
37	24"
38	
39	app:navigationIconTint="?attr/colorOnPrimary"/>
40	
41	</com.google.android.material.appbar.AppBarLayout>
42	
43	<ScrollView
44	android:layout_width="0dp"
45	android:layout_height="0dp"
46	
47	app:layout_constraintTop_toBottomOf="@id/topAppBar
48	Layout"
49	
50	app:layout_constraintBottom_toBottomOf="parent"
51	
52	app:layout_constraintStart_toStartOf="parent"
53	app:layout_constraintEnd_toEndOf="parent">
54	
55	

56	
57	<androidx.constraintlayout.widget.ConstraintLayout
58	android:layout_width="match_parent"
59	android:layout_height="wrap_content"
60	android:padding="20dp"
61	
62	app:layout_constraintTop_toTopOf="parent"
63	
64	app:layout_constraintStart_toStartOf="parent"
65	
66	app:layout_constraintEnd_toEndOf="parent">
67	
68	
69	<com.google.android.material.imageview.ShapeableIm
70	ageView
71	android:id="@+id/riderImage"
72	android:layout_width="300dp"
73	android:layout_height="400dp"
74	android:scaleType="centerCrop"
75	
76	app:shapeAppearanceOverlay="@style/RoundedImageVie
77	w"
78	
79	app:layout_constraintTop_toTopOf="parent"
80	
81	app:layout_constraintStart_toStartOf="parent"
82	
83	app:layout_constraintEnd_toEndOf="parent"
84	android:src="@drawable/kr_geats"/>
85	

86	<TextView
87	android:id="@+id/riderName"
88	
89	android:layout_width="wrap_content"
90	
91	android:layout_height="wrap_content"
92	android:layout_marginTop="15dp"
93	
94	app:layout_constraintTop_toBottomOf="@id/riderImage"
95	"
96	
97	app:layout_constraintStart_toStartOf="parent"
98	
99	app:layout_constraintEnd_toEndOf="parent"
100	tools:text="Kamen Rider Gotchard"
101	
102	android:gravity="center_horizontal"
103	android:textSize="35sp"
104	android:textStyle="bold"/>
105	
106	<TextView
107	android:id="@+id/riderYear"
108	android:layout_width="0dp"
109	
110	android:layout_height="wrap_content"
	android:layout_marginTop="20dp"
	app:layout_constraintTop_toBottomOf="@id/riderName"
	"

	<pre> app:layout_constraintStart_toStartOf="parent" app:layout_constraintEnd_toEndOf="parent" tools:text="Tahun: 2022" android:textSize="22sp" android:textStyle="bold"/> <TextView android:id="@+id/descTitle" android:layout_width="match_parent" android:layout_height="wrap_content" android:layout_marginTop="15dp" app:layout_constraintTop_toBottomOf="@id/riderYear " app:layout_constraintStart_toStartOf="parent" app:layout_constraintEnd_toEndOf="parent" android:text="@string/desc" android:textSize="22sp" android:textStyle="bold"/> <TextView android:id="@+id/descBody" android:layout_width="match_parent" </pre>
--	---

	<pre> android:layout_height="wrap_content" android:layout_marginTop="7dp" app:layout_constraintTop_toBottomOf="@id/descTitle " app:layout_constraintStart_toStartOf="parent" app:layout_constraintEnd_toEndOf="parent" tools:text="Deskripsi panjang lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum" android:textSize="20sp" android:justificationMode="inter_word"/> </androidx.constraintlayout.widget.ConstraintLayout> </ScrollView> </androidx.constraintlayout.widget.ConstraintLayout> </pre>
--	--

fragment_detail.xml (landscape)

Tabel 24. Source Code XML fragment_detail.xml (landscape)

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

2	<androidx.constraintlayout.widget.ConstraintLayout
3	
4	xmlns:android="http://schemas.android.com/apk/res/
5	android"
6	xmlns:app="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="match_parent"
11	tools:context=".DetailFragment">
12	
13	
14	<com.google.android.material.appbar.AppBarLayout
15	android:id="@+id/topAppBarLayout"
16	android:layout_width="0dp"
17	android:layout_height="wrap_content"
18	android:background="?attr/colorPrimary"
19	app:layout_constraintTop_toTopOf="parent"
20	
21	app:layout_constraintStart_toStartOf="parent"
22	app:layout_constraintEnd_toEndOf="parent">
23	
24	
25	<com.google.android.material.appbar.MaterialToolba
26	r
27	android:id="@+id/topAppBar"
28	android:layout_width="match_parent"
29	android:layout_height="match_parent"
30	app:title="Detail"
31	

32	
33	app:titleTextColor="?attr/colorOnPrimary"
34	
35	app:navigationIcon="@drawable/baseline_arrow_back_
36	24"
37	
38	app:navigationIconTint="?attr/colorOnPrimary"/>
39	
40	</com.google.android.material.appbar.AppBarLayout>
41	
42	<ScrollView
43	android:layout_width="0dp"
44	android:layout_height="0dp"
45	
46	app:layout_constraintTop_toBottomOf="@id/topAppBar
47	Layout"
48	
49	app:layout_constraintBottom_toBottomOf="parent"
50	
51	app:layout_constraintStart_toStartOf="parent"
52	app:layout_constraintEnd_toEndOf="parent">
53	
54	
55	<androidx.constraintlayout.widget.ConstraintLayout
56	android:layout_width="match_parent"
57	android:layout_height="wrap_content"
58	android:padding="25dp"
59	
60	app:layout_constraintTop_toTopOf="parent"
61	

62	
63	app:layout_constraintStart_toStartOf="parent"
64	
65	app:layout_constraintEnd_toEndOf="parent">
66	
67	
68	<com.google.android.material.imageview.ShapeableImage
69	View
70	android:id="@+id/riderImage"
71	android:layout_width="240dp"
72	android:layout_height="300dp"
73	android:scaleType="centerCrop"
74	
75	app:shapeAppearanceOverlay="@style/RoundedImageVie
76	w"
77	
78	app:layout_constraintTop_toTopOf="parent"
79	
80	app:layout_constraintBottom_toBottomOf="parent"
81	
82	app:layout_constraintStart_toStartOf="parent"
83	
84	app:layout_constraintEnd_toStartOf="@id/infoText"
85	android:src="@drawable/kr_geats"/>
86	
87	
88	<androidx.constraintlayout.widget.ConstraintLayout
89	android:id="@+id/infoText"
90	android:layout_width="0dp"
91	

92	
93	android:layout_height="wrap_content"
94	android:layout_marginStart="20dp"
95	
96	app:layout_constraintTop_toTopOf="parent"
97	
98	app:layout_constraintStart_toEndOf="@id/riderImage
99	"
100	
101	app:layout_constraintEnd_toEndOf="parent">
102	
103	<TextView
104	android:id="@+id/riderName"
105	
106	android:layout_width="wrap_content"
107	
108	android:layout_height="wrap_content"
109	
110	app:layout_constraintTop_toTopOf="parent"
111	
112	app:layout_constraintStart_toStartOf="parent"
113	tools:text="Kamen Rider Geats"
114	android:textSize="40sp"
115	android:textStyle="bold"/>
116	
117	<TextView
118	android:id="@+id/riderYear"
119	android:layout_width="0dp"
120	
	android:layout_height="wrap_content"

	<pre> android:layout_marginTop="15dp" app:layout_constraintTop_toBottomOf="@id/riderName " app:layout_constraintStart_toStartOf="parent" app:layout_constraintEnd_toEndOf="parent" tools:text="Tahun: 2022" android:textSize="25sp" android:textStyle="bold"/> <TextView android:id="@+id/descTitle" android:layout_width="match_parent" android:layout_height="wrap_content" android:layout_marginTop="10dp" app:layout_constraintTop_toBottomOf="@id/riderYear " app:layout_constraintStart_toStartOf="parent" android:text="@string/desc" android:textSize="25sp" android:textStyle="bold"/> </pre>
--	--

	<pre> <TextView android:id="@+id/descBody" android:layout_width="match_parent" android:layout_height="wrap_content" android:layout_marginTop="7dp" app:layout_constraintTop_toBottomOf="@id/descTitle " app:layout_constraintStart_toStartOf="parent" app:layout_constraintEnd_toEndOf="parent" tools:text="Deskripsi panjang lorem ipsum lorem ipsum lorem ipsum lorem ipsum lorem ipsum" android:textSize="22sp" android:justificationMode="inter_word"/> </androidx.constraintlayout.widget.ConstraintLayout> </androidx.constraintlayout.widget.ConstraintLayout> </pre>
--	--

	<pre> </ScrollView> </androidx.constraintlayout.widget.ConstraintLayout> </pre>
--	--

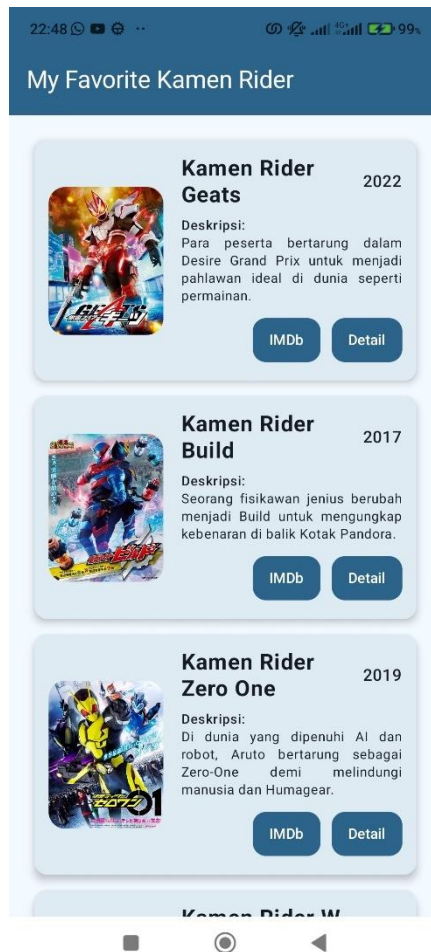
main_graph.xml

Tabel 25. Source Code XML main_graph.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<navigation
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	xmlns:app="http://schemas.android.com/apk/res-auto"
5	xmlns:tools="http://schemas.android.com/tools"
	android:id="@+id/main_graph"
6	app:startDestination="@id/listFragment">
7	
8	<fragment
9	android:id="@+id/detailFragment"
10	android:name="com.example.scrollablexml.DetailFragment"
11	android:label="fragment_detail"
12	tools:layout="@layout/fragment_detail" >
13	<argument
14	android:name="riderId"
15	app:argType="integer" />
16	</fragment>

17	<fragment
18	android:id="@+id/listFragment"
19	android:name="com.example.scrollablexml.ListFragment"
20	android:label="fragment_list"
21	tools:layout="@layout/fragment_list" >
22	<action
23	android:id="@+id/action_listFragment_to_detailFragment"
24	app:destination="@id/detailFragment" />
25	</fragment>
26	</navigation>

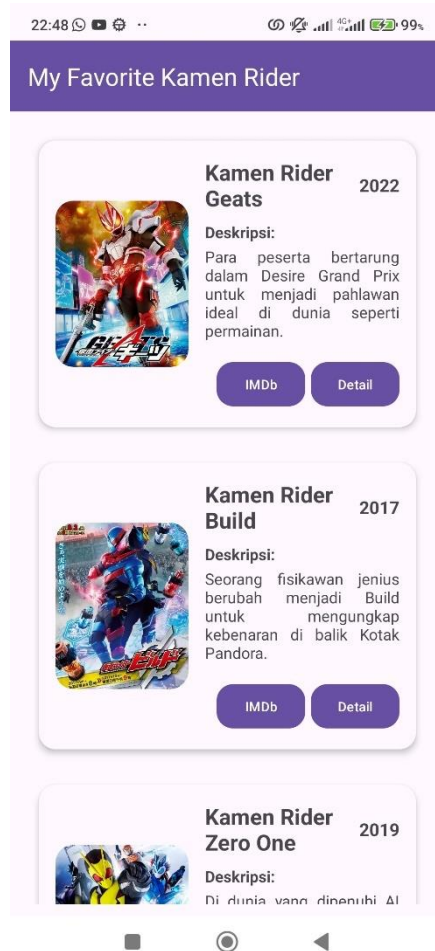
B. Output Program



Gambar 13. Screenshot List Screen Soal 1 Versi Jetpack Compose



Gambar 14. Screenshot Detail Screen Soal 1 Versi Jetpack Compose



Gambar 15. Screenshot List Screen Soal 1 Versi XML



Gambar 16. Screenshot Detail Screen Soal 1 Versi XML

C. Pembahasan

1) Versi Jetpack Compose

MainActivity.kt

Pada MainActivity.kt, kode ini membangun aplikasi Android berbasis Jetpack Compose yang menggunakan sistem navigasi antar layar. Di dalam MainActivity, fungsi `enableEdgeToEdge()` digunakan untuk mengaktifkan tampilan layar penuh, kemudian UI diatur menggunakan `setContent` yang membungkus aplikasi dengan tema `ScrollableComposeTheme`. Fungsi `MyFavKamenRiderApp()` dipanggil untuk

mengatur navigasi antar komponen menggunakan NavHost dan rememberNavController. Navigasi dimulai dari ListScreen, dan saat item dipilih, akan diarahkan ke DetailScreen dengan parameter id yang diambil dari rute. Jika id tidak valid, maka nilai default-nya adalah 0.

Card.kt

Pada Card.kt, terdapat komponen UI yang mendefinisikan dua tampilan berbeda berdasarkan orientasi perangkat: portrait dan landscape. Fungsi utama di sini adalah RiderCard, yang menerima objek KamenRider, NavController, dan mode orientasi sebagai parameter. Berdasarkan mode orientasi perangkat, RiderCard akan memanggil salah satu dari dua fungsi komposabel: RiderCardPortrait untuk tampilan portrait dan RiderCardLandscape untuk tampilan landscape.

Fungsi RiderCardPortrait dan RiderCardLandscape keduanya mengimplementasikan sebuah Card yang berisi informasi mengenai Kamen Rider. Pada tampilan portrait, informasi ditampilkan dalam format yang lebih vertikal, dengan gambar poster di sebelah kiri dan teks deskripsi di sebelah kanan. Pada tampilan landscape, pengaturan layout disesuaikan agar lebih horizontal dengan gambar yang lebih besar, serta teks yang lebih luas dan lebih banyak ruang untuk penjelasan.

Kedua tampilan ini memiliki komponen yang sama, seperti nama rider, tahun tayang, deskripsi, dan dua tombol. Tombol pertama mengarahkan pengguna ke halaman IMDb Kamen Rider yang bersangkutan, sementara tombol kedua mengarahkan ke halaman detail menggunakan NavController. Desain ini memastikan pengalaman pengguna yang optimal dengan menyesuaikan tata letak dan ukuran elemen UI berdasarkan orientasi layar perangkat yang digunakan.

ListScreen.kt

File `ListScreen.kt` berfungsi sebagai tampilan utama aplikasi yang menampilkan daftar karakter Kamen Rider menggunakan `LazyColumn`. Fungsi `ListScreen` menerima `NavController` sebagai parameter untuk mendukung navigasi ke tampilan detail ketika salah satu item dipilih. Dalam komponen ini, `LocalConfiguration` digunakan untuk mendapatkan konfigurasi layar saat ini, terutama orientasi perangkat (potret atau lanskap) yang disimpan ke dalam variabel `orientationMode` untuk kemudian diteruskan ke setiap kartu rider agar tampilannya bisa disesuaikan jika dibutuhkan.

Struktur visual utama dibangun dengan `Scaffold`, yang memiliki `TopAppBar` sebagai header dan daftar rider sebagai isi. `AppBar` ini menampilkan judul yang diambil dari `stringResource`, dan menggunakan skema warna dari `MaterialTheme` untuk tampilan yang konsisten. Daftar Kamen Rider ditampilkan dalam `LazyColumn` yang memungkinkan scroll vertikal dan menggunakan `items` untuk menghasilkan item secara efisien. Masing-masing item menggunakan komponen `RiderCard` (yang tidak ditampilkan dalam file ini) dan diberikan `NavController` serta informasi orientasi sebagai parameter, memungkinkan setiap kartu untuk merespons interaksi pengguna, seperti klik untuk melihat detail.

Selain itu, dua fungsi `@Preview` disediakan untuk melihat pratinjau tampilan daftar pada dua mode orientasi layar: potret dan lanskap. Ini membantu pengembang memastikan bahwa tampilan tetap responsif dan nyaman dilihat di berbagai ukuran dan arah layar. Dengan demikian, file ini menjadi pusat navigasi awal dalam aplikasi, menyajikan daftar Kamen Rider secara bersih, adaptif, dan siap untuk berinteraksi.

DetailScreen.kt

Pada `DetailScreen.kt`, file ini membentuk tampilan detail dari setiap karakter Kamen Rider yang dipilih dari daftar utama. Fungsi utama `DetailScreen` menerima `NavController` dan `id` untuk mencari data `KamenRider` berdasarkan ID dari daftar

yang tersedia. Berdasarkan orientasi perangkat yang dideteksi dengan `LocalConfiguration`, tampilan akan disesuaikan menjadi potret atau lanskap menggunakan dua layout terpisah: `DetailPortraitLayout` dan `DetailLandscapeLayout`.

Jika orientasi perangkat dalam mode potret, layout menampilkan gambar rider dalam ukuran besar di atas, lalu diikuti nama, tahun, dan deskripsi dalam susunan kolom vertikal yang dapat discroll. Sedangkan pada mode lanskap, layout menggunakan `Row` untuk menyusun gambar di sisi kiri dan informasi di sisi kanan secara horizontal, agar lebih optimal memanfaatkan lebar layar. Gambar menggunakan `Image` dengan efek rounded corner, dan teks ditata rapi menggunakan `Text`, `Spacer`, dan pengaturan font yang menonjolkan judul serta informasi penting.

Navigasi balik ke halaman sebelumnya disediakan oleh `AppBar` dengan ikon panah kembali. Jika data rider tidak ditemukan (misalnya karena ID tidak valid), akan ditampilkan teks "Data not found". Terakhir, disediakan juga dua fungsi `@Preview` untuk menampilkan pratinjau tampilan secara langsung di Android Studio, baik untuk mode potret maupun lanskap. Dengan struktur ini, aplikasi memberikan pengalaman visual yang fleksibel dan responsif untuk berbagai orientasi layar.

Routes.kt

Pada `Routes.kt`, didefinisikan sebuah objek `Routes` yang berfungsi untuk menyimpan konstanta yang digunakan dalam navigasi antar layar pada aplikasi. Objek ini menyimpan dua properti: `listScreen` dan `detailScreen`, yang masing-masing mewakili rute untuk layar daftar dan layar detail dalam aplikasi. Rute `listScreen` bertujuan untuk menavigasi ke layar yang menampilkan daftar item, sementara `detailScreen` digunakan untuk menavigasi ke layar yang menampilkan detail dari item yang dipilih. Dengan mendefinisikan rute-rute ini dalam satu

tempat, aplikasi menjadi lebih terorganisir dan memudahkan pengelolaan navigasi, karena jika ada perubahan dalam nama rute, hanya perlu mengubahnya di file Routes.kt saja tanpa harus mencarinya di seluruh proyek. Hal ini juga membuat kode lebih mudah dipelihara dan mengurangi kemungkinan kesalahan saat menggunakan rute dalam aplikasi.

KamenRider.kt

Pada KamenRider.kt, didefinisikan sebuah kelas data KamenRider yang digunakan untuk merepresentasikan data mengenai karakter Kamen Rider dalam aplikasi. Kelas ini memiliki beberapa properti yang menggambarkan informasi penting tentang Kamen Rider, yaitu: id (tipe data Int) yang berfungsi sebagai identifier unik untuk setiap Kamen Rider, name (tipe data String) yang menyimpan nama Kamen Rider, description (tipe data String) yang menyimpan deskripsi tentang karakter tersebut, year (tipe data Int) yang menyimpan tahun debut Kamen Rider, imdbUrl (tipe data String) yang menyimpan URL ke halaman IMDb Kamen Rider, dan posterRes (tipe data Int) yang menyimpan referensi ke sumber daya gambar poster Kamen Rider. Kelas data ini memungkinkan pemrograman yang lebih efisien karena secara otomatis menyediakan metode untuk membandingkan, menyalin, dan mendeskripsikan objek KamenRider. Dengan menggunakan kelas ini, aplikasi dapat menyimpan dan menampilkan data terkait Kamen Rider dengan struktur yang jelas dan mudah diakses.

KamenRiderList.kt

Pada KamenRiderList.kt, terdapat sebuah fungsi bernama getKamenRiderList() yang mengembalikan daftar (list) objek KamenRider. Fungsi ini berfungsi untuk menyediakan data statis mengenai berbagai karakter Kamen Rider yang ada dalam aplikasi. Setiap objek KamenRider yang dimasukkan dalam daftar berisi informasi tentang id, name, description, year, imdbUrl, dan posterRes. id adalah identifier

unik untuk setiap karakter, name berisi nama Kamen Rider, description menjelaskan latar belakang cerita karakter tersebut, year menunjukkan tahun debut Kamen Rider, imdbUrl adalah link menuju halaman IMDb karakter tersebut, dan posterRes merujuk pada sumber daya gambar poster Kamen Rider yang disertakan dalam aplikasi. Fungsi ini menyusun daftar Kamen Rider yang berbeda, seperti Kamen Rider Geats, Kamen Rider Build, dan lainnya, yang masing-masing memiliki deskripsi dan tahun tayang yang unik. Dengan menggunakan fungsi ini, aplikasi dapat menampilkan daftar Kamen Rider yang terstruktur dengan baik, lengkap dengan gambar dan informasi yang relevan.

2) Versi XML

MainActivity.kt:

Pada MainActivity.kt, terdapat sebuah MainActivity yang merupakan kelas utama untuk aplikasi Android. Kelas ini mewarisi AppCompatActivity, yang merupakan kelas dasar untuk aktivitas yang mendukung fitur-fitur kompatibilitas dengan versi Android lebih lama. Pada metode onCreate(), yang dipanggil saat aktivitas pertama kali dibuat, dipanggil metode super.onCreate(savedInstanceState) untuk memastikan siklus hidup aktivitas berjalan dengan benar. Selanjutnya, metode enableEdgeToEdge() dipanggil untuk memungkinkan tampilan aplikasi menutupi area layar secara penuh, termasuk area tepi (edge-to-edge), memberikan pengalaman visual yang lebih imersif. Setelah itu, setContentView(R.layout.activity_main) digunakan untuk menetapkan tampilan antarmuka pengguna dengan menghubungkan layout XML yang berada dalam file activity_main.xml ke aktivitas ini. Dengan begitu, aplikasi akan menampilkan UI sesuai dengan desain yang didefinisikan dalam XML tersebut saat aktivitas ini dimulai.

MainAdapter.kt

Pada `MainAdapter.kt`, terdapat sebuah kelas `MainAdapter` yang merupakan adapter untuk `RecyclerView`, digunakan untuk menampilkan daftar karakter Kamen Rider dalam tampilan daftar (list) di aplikasi. `MainAdapter` menerima tiga parameter: `listRider` yang berisi daftar objek `KamenRider`, serta dua lambda function `onImdbClick` dan `onDetailClick` yang digunakan untuk menangani aksi ketika tombol IMDb dan Detail diklik. Adapter ini memiliki kelas internal `ViewHolder` yang bertanggung jawab untuk mengikat data ke tampilan individual item. Dalam metode `bind()`, data dari objek `KamenRider` seperti gambar, nama, tahun, dan deskripsi ditampilkan di tampilan item menggunakan `AdapterManagerBinding`, yang memetakan elemen UI di XML ke objek di kode. Tombol IMDb dan Detail masing-masing diberi listener untuk menanggapi klik, yang akan menjalankan lambda function yang diberikan. Metode `onCreateViewHolder()` digunakan untuk membuat tampilan item baru dengan `AdapterManagerBinding`, sementara `onBindViewHolder()` mengikat data ke tampilan untuk setiap posisi dalam daftar. Metode `getItemCount()` mengembalikan jumlah item dalam `listRider`. Adapter ini memudahkan pengelolaan dan penampilan data dalam format yang terstruktur di `RecyclerView`.

ListFragment.kt

Pada `ListFragment.kt`, terdapat sebuah fragment yang bertugas untuk menampilkan daftar Kamen Rider dalam bentuk `RecyclerView`. Di dalam fragment ini, terdapat binding dengan layout `FragmentListBinding` yang memungkinkan pengikatan elemen UI ke dalam kode. Pada `onCreateView()`, fragment ini menginisialisasi `MainAdapter` yang digunakan untuk menampilkan daftar `KamenRider` dengan memanfaatkan data yang diperoleh dari fungsi `getKamenRiderList()`. Adapter ini menerima dua fungsi lambda, yaitu `onDetailClick` yang akan menavigasi ke fragment detail menggunakan `NavController`, dan `onImdbClick` yang membuka URL IMDb di aplikasi browser menggunakan `Intent`. Ketika item di daftar dipilih, aksi untuk menavigasi ke fragment lain dilakukan dengan

ListFragmentDirections.actionListFragmentToDetailFragment(riderId). Adapter kemudian dihubungkan dengan RecyclerView melalui binding, dan data ditampilkan sesuai dengan desain. onDestroyView() digunakan untuk membersihkan referensi binding setelah tampilan fragment dihancurkan. Selain itu, terdapat metode newInstance() yang memungkinkan pembuatan instance fragment dengan argumen tertentu. Fragment ini bertanggung jawab untuk menampilkan daftar Kamen Rider dan menangani interaksi pengguna seperti navigasi dan membuka IMDb.

DetailFragment.kt

Pada DetailFragment.kt, fragment ini bertugas untuk menampilkan detail informasi mengenai Kamen Rider yang dipilih pengguna dari daftar. Fragment ini menggunakan ViewBinding dengan mengikat elemen UI dari layout FragmentDetailBinding. Pada onCreateView(), fragment mendapatkan parameter riderId yang diteruskan melalui navArgs(), yang digunakan untuk mencari data Kamen Rider sesuai dengan ID yang diberikan. Fungsi getKamenRiderList() digunakan untuk memperoleh daftar Kamen Rider, dan kemudian mencari item yang cocok dengan ID tersebut. Jika data ditemukan, informasi seperti poster, nama, tahun, dan deskripsi Kamen Rider ditampilkan pada tampilan menggunakan binding. Selain itu, setNavigationOnClickListener pada topAppBar digunakan untuk menangani aksi ketika pengguna menekan tombol kembali di bagian atas aplikasi, yang akan memicu aksi onBackPressed() untuk kembali ke halaman sebelumnya. Pada onDestroyView(), referensi _binding dibersihkan untuk mencegah memory leak setelah tampilan fragment dihancurkan. Fragment ini memastikan pengguna dapat melihat informasi detail mengenai Kamen Rider yang dipilih.

KamenRider.kt

Pada `KamenRider.kt`, didefinisikan sebuah kelas data `KamenRider` yang digunakan untuk merepresentasikan data mengenai karakter Kamen Rider dalam aplikasi. Kelas ini memiliki beberapa properti yang menggambarkan informasi penting tentang Kamen Rider, yaitu: `id` (tipe data `Int`) yang berfungsi sebagai identifier unik untuk setiap Kamen Rider, `name` (tipe data `String`) yang menyimpan nama Kamen Rider, `description` (tipe data `String`) yang menyimpan deskripsi tentang karakter tersebut, `year` (tipe data `Int`) yang menyimpan tahun debut Kamen Rider, `imdbUrl` (tipe data `String`) yang menyimpan URL ke halaman IMDb Kamen Rider, dan `posterRes` (tipe data `Int`) yang menyimpan referensi ke sumber daya gambar poster Kamen Rider. Kelas data ini memungkinkan pemrograman yang lebih efisien karena secara otomatis menyediakan metode untuk membandingkan, menyalin, dan mendeskripsikan objek `KamenRider`. Dengan menggunakan kelas ini, aplikasi dapat menyimpan dan menampilkan data terkait Kamen Rider dengan struktur yang jelas dan mudah diakses.

`KamenRiderList.kt`

Pada `KamenRiderList.kt`, terdapat sebuah fungsi bernama `getKamenRiderList()` yang mengembalikan daftar (list) objek `KamenRider`. Fungsi ini berfungsi untuk menyediakan data statis mengenai berbagai karakter Kamen Rider yang ada dalam aplikasi. Setiap objek `KamenRider` yang dimasukkan dalam daftar berisi informasi tentang `id`, `name`, `description`, `year`, `imdbUrl`, dan `posterRes`. `id` adalah identifier unik untuk setiap karakter, `name` berisi nama Kamen Rider, `description` menjelaskan latar belakang cerita karakter tersebut, `year` menunjukkan tahun debut Kamen Rider, `imdbUrl` adalah link menuju halaman IMDb karakter tersebut, dan `posterRes` merujuk pada sumber daya gambar poster Kamen Rider yang disertakan dalam aplikasi. Fungsi ini menyusun daftar Kamen Rider yang berbeda, seperti Kamen Rider Geats, Kamen Rider Build, dan lainnya, yang masing-masing memiliki deskripsi dan tahun tayang yang unik. Dengan menggunakan fungsi ini,

aplikasi dapat menampilkan daftar Kamen Rider yang terstruktur dengan baik, lengkap dengan gambar dan informasi yang relevan.

activity_main.xml:

Pada activity_main.xml, layout ini menggunakan ConstraintLayout sebagai root view untuk mendefinisikan tampilan aplikasi. ConstraintLayout memberikan fleksibilitas dalam menentukan aturan tata letak dengan menggunakan constraint pada posisi elemen-elemen di dalamnya. Di dalam layout ini, terdapat FragmentContainerView yang digunakan untuk menampilkan fragment dengan ID fragmentContainerView6. Elemen ini berfungsi sebagai wadah untuk NavHostFragment, yang berperan sebagai pengontrol navigasi antar fragment dalam aplikasi. Dengan menetapkan atribut app:navGraph ke @navigation/main_graph, NavHostFragment dihubungkan dengan file graf navigasi main_graph, yang mendefinisikan jalur navigasi antara fragment. Atribut app:defaultNavHost="true" menunjukkan bahwa NavHostFragment akan menangani navigasi kembali secara default, menjadikannya sebagai fragment utama yang dapat mengelola transisi antar fragment. Layout ini memastikan bahwa FragmentContainerView mengisi seluruh ruang layar, dengan mengikatnya ke sisi atas, bawah, kiri, dan kanan dari parent container menggunakan constraints. Selain itu, atribut android:fitsSystemWindows="true" memastikan bahwa layout dapat beradaptasi dengan elemen sistem, seperti status bar atau navigasi bar, dengan memberikan ruang yang cukup pada bagian atas dan bawah.

adapter_main.xml

Pada adapter_main.xml, layout ini dirancang untuk menampilkan elemen-elemen informasi mengenai Kamen Rider dalam bentuk yang terstruktur dan menarik, menggunakan ConstraintLayout sebagai root view. Di dalamnya terdapat CardView, yang berfungsi untuk memberikan efek elevasi dan sudut melengkung

pada konten, memberikan tampilan yang lebih modern dan terpisah dengan jelas dari latar belakang. CardView ini memiliki margin 16dp dan menggunakan warna latar belakang dari atribut sistem (?android:attr/colorBackground). Di dalam CardView, terdapat ConstraintLayout lagi yang berfungsi untuk menata elemen-elemen UI, dengan padding 15dp untuk memberikan ruang antara konten dan tepi layout.

Gambar Kamen Rider ditampilkan menggunakan ShapeableImageView dengan lebar 120dp dan tinggi 150dp, serta efek crop di tengah gambar agar tampil dengan proporsi yang tepat. Gambar ini terletak di sisi kiri dan dibatasi oleh constraint yang mengikatnya ke atas, bawah, kiri, dan kanan dari parent layout. Di sebelah kanan gambar, terdapat ConstraintLayout lain yang menampung dua TextView untuk menampilkan nama dan tahun Kamen Rider. Nama Rider memiliki bobot horizontal yang lebih besar (6), sementara tahun ditampilkan dengan bobot lebih kecil (2) dan diatur ke posisi kanan.

Selanjutnya, terdapat dua TextView untuk menampilkan judul dan deskripsi dari Kamen Rider. Deskripsi ditampilkan dalam teks panjang dengan justificationMode="inter_word" untuk memastikan teks diratakan secara merata. Di bagian bawah deskripsi, terdapat ConstraintLayout lainnya yang berisi dua tombol, imdbBtn dan detailBtn, yang masing-masing memiliki ukuran dan radius sudut 15dp, dengan teks yang disesuaikan untuk menampilkan "IMDb" dan "Detail". Tombol-tombol ini diatur untuk berada di posisi bawah dan di sebelah satu sama lain. Dengan desain ini, tampilan kartu Kamen Rider terlihat rapi, terstruktur, dan responsif pada berbagai ukuran layar.

fragment_list.xml

Pada fragment_list.xml, layout ini dirancang untuk menampilkan daftar Kamen Rider menggunakan RecyclerView yang diletakkan di bawah AppBarLayout. Root view menggunakan ConstraintLayout untuk memastikan elemen-elemen dalam

layout dapat diatur dengan constraint yang fleksibel dan responsif pada berbagai ukuran layar. Di bagian atas, terdapat AppBarLayout yang berfungsi untuk menampung MaterialToolbar. Toolbar ini memiliki judul "My Favorite Kamen Rider" dengan warna teks yang disesuaikan menggunakan atribut tema `?attr/colorOnPrimary`, serta latar belakang menggunakan `?attr/colorPrimary`, yang memberikan tampilan yang konsisten dengan tema aplikasi.

Di bawah toolbar, terdapat RecyclerView yang diatur untuk mengisi seluruh sisa ruang di bawah toolbar dengan padding 10dp di sekelilingnya. RecyclerView ini menggunakan LinearLayoutManager untuk menampilkan daftar item secara vertikal dan disesuaikan dengan layout item yang ada di `adapter_main.xml`. Jumlah item dalam daftar ditentukan oleh atribut `tools:itemCount="7"`, memberikan gambaran tentang jumlah item yang akan ditampilkan selama desain. Dengan menggunakan ConstraintLayout, RecyclerView ini terhubung dengan toolbar di bagian atas dan memastikan elemen-elemen tersebut terorganisir dengan baik. Desain ini memberikan tampilan yang bersih dan terstruktur dengan ruang yang cukup untuk menampilkan daftar Kamen Rider secara efisien.

fragment_detail.xml

Pada `fragment_detail.xml`, layout ini dirancang untuk menampilkan detail dari Kamen Rider yang dipilih dalam aplikasi. Root view menggunakan ConstraintLayout untuk pengaturan elemen-elemen yang fleksibel dan responsif. Di bagian atas, terdapat AppBarLayout yang menampung MaterialToolbar dengan judul "Detail" dan ikon navigasi kembali (`@drawable/baseline_arrow_back_24`) yang memungkinkan pengguna untuk kembali ke layar sebelumnya. Warna teks pada toolbar disesuaikan menggunakan tema dengan atribut `?attr/colorOnPrimary`.

Di bawah toolbar, terdapat ScrollView yang memungkinkan konten untuk digulir jika melebihi ukuran layar. Di dalam ScrollView, terdapat ConstraintLayout yang menampung elemen-elemen tampilan seperti gambar, teks, dan deskripsi Kamen

Rider. `ShapeableImageView` digunakan untuk menampilkan gambar poster Kamen Rider dengan bentuk sudut membulat menggunakan `@style/RoundedImageView`. Di bawah gambar, terdapat `TextView` yang menampilkan nama Kamen Rider, diikuti dengan `TextView` lainnya yang menunjukkan tahun rilis, keduanya dengan pengaturan teks yang besar dan tebal untuk memperjelas informasi.

Selanjutnya, terdapat `TextView` untuk judul deskripsi yang diikuti dengan `TextView` lain yang menampilkan isi deskripsi Kamen Rider. Semua elemen teks ini diatur dengan margin dan jarak yang cukup agar tampak rapi dan mudah dibaca. Desain ini memberikan tampilan yang terstruktur dan informatif, memungkinkan pengguna untuk melihat detail Kamen Rider dengan jelas, dengan fungsionalitas scroll yang mendukung tampilan konten yang lebih panjang.

fragment_detail.xml (landscape)

Pada `fragment_detail.xml` versi landscape ini, layout dirancang untuk menampilkan detail Kamen Rider dalam orientasi landscape dengan penataan elemen yang lebih horizontal. Layout utama menggunakan `ConstraintLayout` sebagai root view, dengan `AppBarLayout` yang berisi `MaterialToolbar` di bagian atas, yang mencakup judul "Detail" dan ikon navigasi kembali. Toolbar ini memiliki warna teks disesuaikan dengan tema menggunakan `?attr/colorOnPrimary`.

Di bawah toolbar, terdapat `ScrollView` yang memungkinkan tampilan untuk digulir apabila konten melebihi layar. Dalam `ScrollView`, terdapat `ConstraintLayout` dengan elemen-elemen yang disusun untuk memanfaatkan ruang lebih luas dalam orientasi landscape. `ShapeableImageView` digunakan untuk menampilkan gambar Kamen Rider dengan proporsi yang lebih kecil (240dp x 300dp) dan dengan bentuk sudut membulat. Gambar ini diletakkan di sisi kiri, sementara informasi teks, seperti nama Kamen Rider, tahun rilis, judul deskripsi, dan isi deskripsi, ditempatkan di sebelah kanan gambar dalam `ConstraintLayout` terpisah.

Pada bagian kanan, elemen-elemen teks disusun vertikal, dengan TextView untuk nama Kamen Rider yang memiliki ukuran teks besar (40sp) dan tebal, diikuti oleh teks tahun rilis yang sedikit lebih kecil, dan judul serta deskripsi yang memberikan informasi lebih mendalam. Setiap TextView memiliki margin dan jarak antar elemen agar tampilan terlihat rapi dan mudah dibaca. Pengaturan layout ini dirancang untuk menyesuaikan tampilan yang lebih luas dan horizontal, memanfaatkan ruang secara optimal dalam orientasi landscape.

main_graph.xml

Pada `main_graph.xml`, file ini merupakan definisi Navigation Graph yang digunakan untuk mengelola navigasi antar fragment dalam aplikasi Android. Di dalamnya, terdapat dua fragment yang didefinisikan, yaitu `listFragment` dan `detailFragment`. `listFragment` merupakan fragment pertama yang muncul sebagai tampilan awal (start destination) dari aplikasi, yang ditandai dengan atribut `app:startDestination="@id/listFragment"`. Fragment ini akan menampilkan daftar Kamen Rider, yang disesuaikan dengan layout `fragment_list.xml`.

`detailFragment` adalah fragment kedua yang digunakan untuk menampilkan detail informasi Kamen Rider tertentu. Fragment ini memiliki satu argument, yaitu `riderId`, yang bertipe integer, yang akan diteruskan saat navigasi dari `listFragment` ke `detailFragment`. Argument ini dapat digunakan untuk menampilkan data spesifik berdasarkan Kamen Rider yang dipilih. Di dalam fragment ini, layout yang digunakan adalah `fragment_detail.xml`.

Untuk menghubungkan kedua fragment, terdapat sebuah action yang memungkinkan navigasi dari `listFragment` ke `detailFragment`. Action ini memiliki ID `action_listFragment_to_detailFragment` dan mendefinisikan bahwa tujuan navigasi adalah `detailFragment`. Dengan demikian, saat pengguna memilih suatu item dari `listFragment`, aplikasi akan menavigasi ke `detailFragment` dan menampilkan informasi yang lebih detail berdasarkan ID yang diteruskan.

SOAL 2

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

Jawab:

Karena RecyclerView lebih fleksibel dan lebih leluasa dalam melakukan kustomisasi terhadap perilaku, tampilan, dan animasi elemen di dalam daftar yang ditampilkan. Selain itu, RecyclerView juga lebih unggul dalam segi performa karena RecyclerView mengoptimalkan memori dengan hanya membuat tampilan yang terlihat oleh pengguna (view recycling), ini membuatnya sangat efisien untuk aplikasi dengan data yang besar.

Jadi, meskipun LazyColumn menawarkan kemudahan dalam penggunaan dan sintaksis yang lebih bersih, RecyclerView masih tetap relevan karena lebih fleksibel, kompatibel dengan berbagai versi Android, dan lebih kuat dalam menangani skenario yang lebih kompleks.

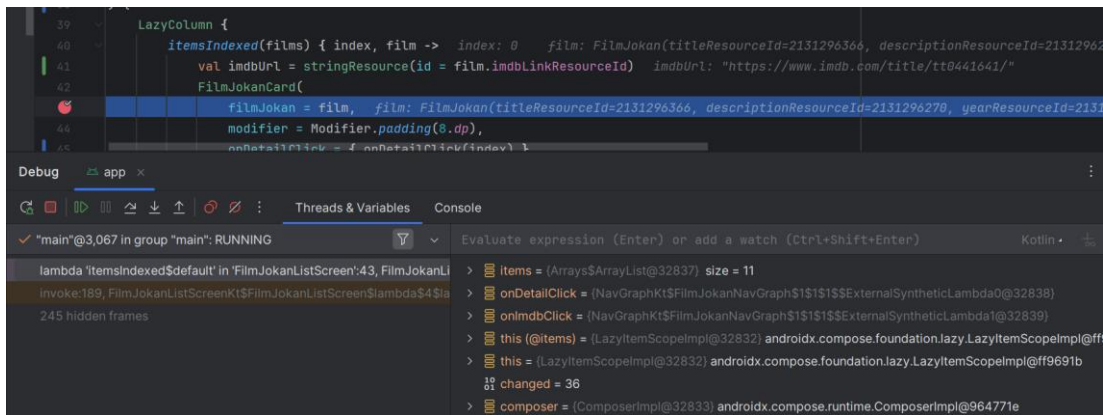
MODUL 4 : VIEWMODEL AND DEBUGGING

SOAL 1

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

- a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
- b. Gunakan ViewModelFactory untuk membuat parameter dengan tipe data String di dalam ViewModel
- c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
- d. Install dan gunakan library Timber untuk logging event berikut:
 - a. Log saat data item masuk ke dalam list
 - b. Log saat tombol Detail dan tombol Explicit Intent ditekan
 - c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
- e. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi.

Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out



Gambar 17. Contoh Penggunaan Debugger

A. Source Code

1) Versi Jetpack Compose

Card.kt

Tabel 26. Source Code Jetpack Compose Card.kt Soal 1

1	package com.example.scrollablecompose.screens
2	
3	import android.content.Context
4	import android.content.Intent
5	import android.net.Uri
6	import androidx.compose.foundation.Image
7	import
8	androidx.compose.foundation.layout.Arrangement
9	import androidx.compose.foundation.layout.Column
10	import
11	androidx.compose.foundation.layout.PaddingValues
12	import androidx.compose.foundation.layout.Row
13	import androidx.compose.foundation.layout.Spacer
14	import
15	androidx.compose.foundation.layout.fillMaxWidth
16	import androidx.compose.foundation.layout.height

17	import androidx.compose.foundation.layout.padding
18	import androidx.compose.foundation.layout.size
19	import androidx.compose.foundation.layout.width
20	import
21	androidx.compose.foundation.layout.wrapContentHeig
22	ht
23	import
24	androidx.compose.foundation.shape.RoundedCornerSha
25	pe
26	import androidx.compose.material3.Button
27	import androidx.compose.material3.Card
28	import androidx.compose.material3.CardDefaults
29	import androidx.compose.material3.MaterialTheme
30	import androidx.compose.material3.Text
31	import androidx.compose.runtime.Composable
32	import androidx.compose.ui.Alignment
33	import androidx.compose.ui.Modifier
34	import androidx.compose.ui.draw.clip
35	import androidx.compose.ui.layout.ContentScale
36	import androidx.compose.ui.res.painterResource
37	import androidx.compose.ui.res.stringResource
38	import androidx.compose.ui.text.font.FontWeight
39	import androidx.compose.ui.text.style.TextAlign
40	import androidx.compose.ui.unit.dp
41	import androidx.compose.ui.unit.sp
42	import com.example.scrollablecompose.R
43	import
44	com.example.scrollablecompose.model.KamenRider
45	import timber.log.Timber
46	

47	@Composable
48	fun RiderCardPortrait(
49	rider: KamenRider,
50	context: Context,
51	onClick: () -> Unit
52){
53	Card(
54	modifier = Modifier
55	.padding(7.dp)
56	.fillMaxWidth()
57	.wrapContentHeight(),
58	shape = MaterialTheme.shapes.medium,
59	colors = CardDefaults.cardColors(
60	containerColor
61	MaterialTheme.colorScheme.surface
62),
63	elevation = CardDefaults.cardElevation(
64	defaultElevation = 5.dp
65)
66) {
67	Row(
68	verticalAlignment = Alignment.Top,
69	modifier = Modifier.padding(5.dp)
70) {
71	Image(
72	painter = painterResource(id =
73	rider.posterRes),
74	contentDescription = "Poster
75	\${rider.name}",
76	modifier = Modifier

77	.size(width = 120.dp, height =
78	150.dp)
79	.padding(8.dp)
80	
81	.align(Alignment.CenterVertically)
82	
83	.clip(RoundedCornerShape(15.dp)),
84	contentScale =
85	ContentScale.FillBounds,
86)
87	Column(Modifier.padding(8.dp)) {
88	Row (
89	modifier = Modifier
90	.fillMaxWidth()
91) {
92	Text (
93	text = rider.name,
94	fontSize = 20.sp,
95	fontWeight =
96	FontWeight.Bold,
97	color =
98	MaterialTheme.colorScheme.onSurface,
99	modifier = Modifier
100	
101	.align(Alignment.CenterVertically)
102	.weight(1f)
103)
104	Spacer(Modifier.width(20.dp))
105	Text (
106	

107	text	=
108	rider.year.toString(),	
109	style	=
110	MaterialTheme.typography.labelMedium,	
111	fontSize = 15.sp,	
112	color	=
113	MaterialTheme.colorScheme.onSurface,	
114	modifier	=
115	Modifier.align(Alignment.CenterVertically)	
116)	
117	}	
118		
119	Spacer(Modifier.height(8.dp))	
120	Text(
121	text	=
122	stringResource(R.string.description),	
123	style	=
124	MaterialTheme.typography.labelMedium,	
125	color	=
126	MaterialTheme.colorScheme.onSurface	
127)	
128	Text(
129	text = rider.description,	
130	style	=
131	MaterialTheme.typography.bodySmall,	
132	color	=
133	MaterialTheme.colorScheme.onSurface,	
134	textAlign = TextAlign.Justify	
135)	
136	Spacer(Modifier.height(8.dp))	

137	
138	Row (
139	Modifier.fillMaxWidth(),
140	horizontalArrangement =
141	Arrangement.End
142) {
143	Button(
144	shape =
145	RoundedCornerShape(12.dp),
146	contentPadding =
147	PaddingValues(horizontal = 15.dp),
148	onClick = {
149	
150	Timber.tag("ListScreen").i("Tombol Explicit Intent
151	ditekan untuk: \${rider.name}, url:
152	\${rider.imdbUrl}")
153	val intent =
154	Intent(Intent.ACTION_VIEW)
155	intent.data =
156	Uri.parse(rider.imdbUrl)
157	
158	intent.setPackage("com.android.chrome")
159	
160	context.startActivity(intent)
161	}
162) {
163	
164	Text(stringResource(R.string.imdb), fontSize =
165	13.sp)
166	}

167	Spacer (Modifier.width (10.dp))	
168	Button (
169	shape	=
170	RoundedCornerShape (12.dp),	
171	contentPadding	=
172	PaddingValues (horizontal = 15.dp),	
173	onClick = onClick	
174) {	
175		
176	Text (stringResource (R.string.detail),	fontSize =
177	13.sp)	
178	}	
179	}	
180	}	
181	}	
182	}	
183	}	
184		
185	@Composable	
186	fun RiderCardLandscape (
187	rider: KamenRider,	
188	context: Context,	
189	onClick: () -> Unit	
190) {	
191	Card (
192	modifier = Modifier	
193	.padding (7.dp)	
194	.fillMaxWidth ()	
195	.wrapContentHeight (),	
196	shape = MaterialTheme.shapes.medium,	

197	colors = CardDefaults.cardColors(
198	containerColor
199	MaterialTheme.colorScheme.surface
200),
201	elevation
202	CardDefaults.cardElevation(defaultElevation = 5.dp)
203) {
204	Row(
205	verticalAlignment = Alignment.Top,
206	modifier = Modifier.padding(5.dp)
207) {
208	Image(
209	painter = painterResource(id =
210	rider.posterRes),
211	contentDescription = "Poster
212	\${rider.name}",
213	modifier = Modifier
214	.size(width = 192.dp, height =
215	240.dp)
216	.padding(8.dp)
217	
218	.align(Alignment.CenterVertically)
219	
220	.clip(RoundedCornerShape(15.dp)),
221	contentScale
222	ContentScale.FillBounds,
223)
224	Spacer(Modifier.width(5.dp))
225	Column(Modifier.padding(8.dp)) {
226	

227	Row(modifier	=
228	Modifier.fillMaxWidth()) {	
229	Text(
230	text = rider.name,	
231	fontSize = 30.sp,	
232	fontWeight	=
233	FontWeight.Bold,	
234	color	=
235	MaterialTheme.colorScheme.onSurface,	
236	modifier = Modifier	
237		
238	.align(Alignment.CenterVertically)	
239	.weight(1f)	
240)	
241	Spacer(Modifier.width(20.dp))	
242	Text(
243	text	=
	rider.year.toString(),	
	style	=
	MaterialTheme.typography.labelMedium,	
	fontSize = 20.sp,	
	color	=
	MaterialTheme.colorScheme.onSurface,	
	modifier	=
	Modifier.align(Alignment.CenterVertically)	
)	
	}	
	Spacer(Modifier.height(15.dp))	
	Text(

```

        text
        =
stringResource(R.string.description),
        style
        =
MaterialTheme.typography.labelMedium,
        fontSize = 20.sp,
        color
        =
MaterialTheme.colorScheme.onSurface
    )
    Spacer(Modifier.height(5.dp))
    Text(
        text = rider.description,
        style
        =
MaterialTheme.typography.bodyMedium,
        fontSize = 18.sp,
        color
        =
MaterialTheme.colorScheme.onSurface,
        textAlign = TextAlign.Justify
    )

    Spacer(Modifier.height(50.dp))
    Row(
        Modifier.fillMaxWidth(),
        horizontalArrangement
        =
Arrangement.End
    ) {
        Button(
            shape
            =
RoundedCornerShape(16.dp),
            contentPadding
            =
PaddingValues(vertical = 15.dp, horizontal = 20.dp),

```

	<pre> onClick = { Timber.tag("ListScreen").i("Tombol Explicit Intent ditekan untuk: \${rider.name}, url: \${rider.imdbUrl}") val intent = Intent(Intent.ACTION_VIEW) intent.data = Uri.parse(rider.imdbUrl) intent.setPackage("com.android.chrome") context.startActivity(intent) }) { Text(stringResource(R.string.imdb), fontSize = 18.sp) } Spacer(Modifier.width(10.dp)) Button(shape = RoundedCornerShape(16.dp), contentPadding = PaddingValues(vertical = 15.dp, horizontal = 20.dp), onClick = onClick) { </pre>
--	---

	<pre> Text(stringResource(R.string.detail), fontSize = 18.sp) } } } } } </pre>
--	---

ListScreen.kt

Tabel 27. Source Code Jetpack Compose ListScreen.kt Soal 1

1	package com.example.scrollablecompose.screens
2	
3	import android.content.res.Configuration
4	import
5	androidx.compose.foundation.layout.PaddingValues
6	import
7	androidx.compose.foundation.layout.fillMaxWidth
8	import androidx.compose.foundation.layout.padding
9	import androidx.compose.foundation.lazy.LazyColumn
10	import androidx.compose.foundation.lazy.items
11	import
12	androidx.compose.material3.ExperimentalMaterial3Ap
13	i
14	import androidx.compose.material3.MaterialTheme
15	import androidx.compose.material3.Scaffold
16	import androidx.compose.material3.Text
17	import androidx.compose.material3.TopAppBar
18	import androidx.compose.material3.TopAppBarDefaults

19	import androidx.compose.runtime.Composable
20	import androidx.compose.runtime.LaunchedEffect
21	import androidx.compose.runtime.collectAsState
22	import androidx.compose.runtime.getValue
23	import androidx.compose.runtime.remember
24	import androidx.compose.ui.Modifier
25	import
26	androidx.compose.ui.platform.LocalConfiguration
27	import androidx.compose.ui.platform.LocalContext
28	import androidx.compose.ui.res.stringResource
29	import androidx.compose.ui.tooling.preview.Preview
30	import androidx.compose.ui.unit.dp
31	import
32	androidx.lifecycle.viewmodel.compose.viewModel
33	import androidx.navigation.NavController
34	import com.example.scrollablecompose.R
35	import com.example.scrollablecompose.Routes
36	import
37	com.example.scrollablecompose.ui.theme.ScrollableC
38	omposeTheme
39	import
40	com.example.scrollablecompose.viewmodel.RiderListV
41	iewModel
42	import
43	com.example.scrollablecompose.viewmodel.RiderListV
44	iewModelFactory
45	import timber.log.Timber
46	
47	@OptIn(ExperimentalMaterial3Api::class)
48	@Composable

49	fun ListScreen(navController: NavController? =	=
50	null) {	
51	val title	=
52	stringResource(R.string.topappbarr_title)	
53	val factory = remember(title) {	{
54	RiderListViewModelFactory(title) }	}
55	val viewModel: RiderListViewModel	=
56	viewModel(factory = factory, key = title)	
57		
58	val kamenRiderList	by
59	viewModel.riderList.collectAsState()	
60	val clickedRider	by
61	viewModel.onClickRider.collectAsState()	
62		
63	val orientationMode	=
64	LocalConfiguration.current.orientation	
65	val context = LocalContext.current	
66		
67	LaunchedEffect(clickedRider) {	
68	clickedRider?.let { rider ->	
69		
70	navController?.navigate(Routes.detailScreen	+
71	"/\${rider.id}")	
72	viewModel.clearRiderClick()	
73	}	
74	}	
75		
76	Scaffold(
77	topBar = {	
78	TopAppBar(

79	title = { Text(title) },	
80	colors	=
81	TopAppBarDefaults.topAppBarColors(
82	containerColor	=
83	MaterialTheme.colorScheme.primary,	
84	titleContentColor	=
85	MaterialTheme.colorScheme.onPrimary	
86)	
87)	
88	}	
89) { innerPadding ->	
90	LazyColumn(
91	modifier = Modifier	
92	.fillMaxWidth()	
93	.padding(innerPadding),	
94	contentPadding = PaddingValues(16.dp)	
95) {	
96	items(kamenRiderList) { rider ->	
97	val onClick = {	
98		
99	Timber.tag("ListScreen").i("Tombol Detail ditekan	
100	untuk: \${rider.name}")	
101	viewModel.onRiderClick(rider)	
102	}	
103		
104	if (orientationMode ==	
105	Configuration.ORIENTATION_PORTRAIT) {	
106	RiderCardPortrait(rider	=
107	rider, context = context, onClick = onClick)	
108	} else {	

109	RiderCardLandscape(rider	=
110	rider, context = context, onClick = onClick)	
	}	
	}	
	}	
	}	
	@Preview(showBackground = true, widthDp = 360, heightDp = 800) @Composable fun ListScreenPortPreview() { ScrollableComposeTheme { ListScreen() } }	
	//@Preview(// showBackground = true, // widthDp = 800, // heightDp = 360 //) //@Composable //fun ListScreenLandPreview() { // ScrollableComposeTheme { // ListScreen() // } //}	

	//}
--	-----

DetailScreen.kt

Tabel 28. Source Code Jetpack Compose DetailScreen.kt Soal 1

1	package com.example.scrollablecompose.screens
2	
3	import android.content.res.Configuration
4	import androidx.compose.foundation.Image
5	import androidx.compose.foundation.layout.Column
6	import
7	androidx.compose.foundation.layout.PaddingValues
8	import androidx.compose.foundation.layout.Row
9	import androidx.compose.foundation.layout.Spacer
10	import
11	androidx.compose.foundation.layout.fillMaxSize
12	import
13	androidx.compose.foundation.layout.fillMaxWidth
14	import androidx.compose.foundation.layout.height
15	import androidx.compose.foundation.layout.padding
16	import androidx.compose.foundation.layout.size
17	import androidx.compose.foundation.layout.width
18	import
19	androidx.compose.foundation.rememberScrollState
20	import
21	androidx.compose.foundation.shape.RoundedCornerSha
22	pe
23	import androidx.compose.foundation.verticalScroll
24	import androidx.compose.material.icons.Icons
25	
26	

27	import
28	androidx.compose.material.icons.automirrored.filled.ArrowBack
29	
30	import
31	androidx.compose.material3.ExperimentalMaterial3Api
32	
33	import androidx.compose.material3.Icon
34	import androidx.compose.material3.IconButton
35	import androidx.compose.material3.MaterialTheme
36	import androidx.compose.material3.Scaffold
37	import androidx.compose.material3.Text
38	import androidx.compose.material3.TopAppBar
39	import androidx.compose.material3.TopAppBarDefaults
40	import androidx.compose.runtime.Composable
41	import androidx.compose.runtime.collectAsState
42	import androidx.compose.runtime.remember
43	import androidx.compose.ui.Alignment
44	import androidx.compose.ui.Modifier
45	import androidx.compose.ui.draw.clip
46	import androidx.compose.ui.layout.ContentScale
47	import
48	androidx.compose.ui.platform.LocalConfiguration
49	import androidx.compose.ui.res.painterResource
50	import androidx.compose.ui.res.stringResource
51	import androidx.compose.ui.text.font.FontWeight
52	import androidx.compose.ui.text.style.TextAlign
53	import androidx.compose.ui.tooling.preview.Preview
54	import androidx.compose.ui.unit.dp
55	import androidx.compose.ui.unit.sp
56	

57	import
58	androidx.lifecycle.viewmodel.compose.viewModel
59	import androidx.navigation.NavController
60	import com.example.scrollablecompose.R
61	import
62	com.example.scrollablecompose.model.KamenRider
63	import
64	com.example.scrollablecompose.model.KamenRiderRepo
65	sitory.getKamenRiderList
66	import
67	com.example.scrollablecompose.ui.theme.ScrollableC
68	omposeTheme
69	import
70	com.example.scrollablecompose.viewmodel.DetailView
71	Model
72	import
73	com.example.scrollablecompose.viewmodel.DetailView
74	ModelFactory
75	
76	@OptIn(ExperimentalMaterial3Api::class)
77	@Composable
78	fun DetailScreen(navController: NavController? =
79	null, id: Int) {
80	val factory = remember {
81	DetailViewModelFactory(id) }
82	val viewModel: DetailViewModel =
83	viewModel(factory = factory)
84	val riderState =
85	viewModel.rider.collectAsState()
86	val rider = riderState.value

87	val	orientation	=
88	LocalConfiguration.current.orientation		
89			
90	Scaffold(
91	topBar = {		
92	TopAppBar(
93	title	=	{
94	Text(stringResource(R.string.detail)) },		
95	navigationIcon = {		
96	IconButton(onClick	=	{
97	navController?.navigateUp())) {		
98	Icon(
99	imageVector	=	
100	Icons.AutoMirrored.Filled.ArrowBack,		
101	contentDescription	=	
102	"Back"		
103)		
104	}		
105	},		
106	colors	=	
107	TopAppBarDefaults.topAppBarColors(
108	containerColor	=	
109	MaterialTheme.colorScheme.primary,		
110	titleContentColor	=	
111	MaterialTheme.colorScheme.onPrimary,		
112	navigationIconContentColor	=	
113	MaterialTheme.colorScheme.onPrimary		
114)		
115)		
116	}		

```

117         ) { padding ->
118             if (rider != null) {
119                 if (orientation ==
120 Configuration.ORIENTATION_PORTRAIT) {
121                     DetailPortrait(rider, padding)
122                 } else {
123                     DetailLandscape(rider, padding)
124                 }
125             } else {
126                 Text(
127                     text = "Data not found",
128                     fontSize = 25.sp,
129                     modifier =
130 Modifier.padding(padding)
131                 )
132             }
133         }
134     }
135
136
137     @Composable
138     fun DetailPortrait(rider: KamenRider,
139 paddingValues: PaddingValues) {
140         Column(
141             modifier = Modifier
142                 .padding(paddingValues)
143                 .padding(16.dp)
144                 .verticalScroll(rememberScrollState())
145                 .fillMaxSize(),
146

```

```

147         horizontalAlignment =
148         Alignment.CenterHorizontally
149     ) {
150         RiderImage(rider)
151         Spacer(Modifier.height(15.dp))
152         RiderTextInfo(rider)
153     }
154 }
155
156
157 @Composable
158 fun DetailLandscape(rider: KamenRider,
159 paddingValues: PaddingValues) {
160     Row(
161         modifier = Modifier
162             .fillMaxWidth()
163             .verticalScroll(rememberScrollState())
164             .padding(paddingValues)
165             .padding(16.dp)
166     ) {
167         RiderImage(rider,
168 Modifier.align(Alignment.CenterVertically))
169         Spacer(Modifier.width(12.dp))
170         RiderTextInfo(rider)
171     }
172 }
173
174 @Composable
175 fun RiderImage(rider: KamenRider, modifier:
176 Modifier = Modifier) {

```


177	Image (
178	painter = painterResource(id =
179	rider.posterRes),
180	contentDescription = "Poster
181	\${rider.name}",
182	contentScale = ContentScale.FillBounds,
183	modifier = modifier
184	.size(width = 240.dp, height = 320.dp)
185	.clip(RoundedCornerShape(20.dp))
186)
187	}
188	
189	@Composable
190	fun RiderTextInfo(rider: KamenRider) {
191	val orientation =
192	LocalConfiguration.current.orientation
193	val isPortrait = orientation ==
194	Configuration.ORIENTATION_PORTRAIT
195	
196	Column (
197	modifier = Modifier.fillMaxWidth()
198) {
199	Text (
200	text = rider.name,
201	fontSize = 28.sp,
202	fontWeight = FontWeight.Bold,
203	textAlign = if (isPortrait)
204	TextAlign.Center else TextAlign.Start,
	modifier = Modifier.fillMaxWidth()
)

```

        Spacer(Modifier.height(12.dp))
        Text(
            text = stringResource(R.string.year,
rider.year),
            fontSize = 18.sp,
            fontWeight = FontWeight.Medium
        )
        Spacer(Modifier.height(12.dp))
        Text(
            text
=
stringResource(R.string.description),
            fontSize = 18.sp,
            fontWeight = FontWeight.Bold
        )
        Spacer(Modifier.height(4.dp))
        Text(
            text = rider.description,
            textAlign = TextAlign.Justify
        )
    }
}

@Preview(
    showBackground = true,
    widthDp = 390,
    heightDp = 800,
    name = "Redmi Note 13 Portrait"
)

```

	<pre> @Composable fun DetailScreenPortraitPreview() { ScrollableComposeTheme { DetailScreen(null, getKamenRiderList()[0].id) } } //@Preview(// showBackground = true, // widthDp = 800, // heightDp = 390, // name = "Redmi Note 13 Landscape" //) //@Composable //fun DetailScreenLandscapePreview() { // ScrollableComposeTheme { // val navController = rememberNavController() // DetailScreen(navController, getKamenRiderList()[0].id) // } //} </pre>
--	--

RiderListViewModel.kt

Tabel 29. Source Code Jetpack Compose RiderListViewModel.kt Soal 1

1	package com.example.scrollablecompose.viewmodel
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider
5	import androidx.lifecycle.viewModelScope

6	import
7	com.example.scrollablecompose.model.KamenRider
8	import
9	com.example.scrollablecompose.model.KamenRiderRepo
10	sitory
11	import kotlinx.coroutines.flow.MutableStateFlow
12	import kotlinx.coroutines.flow.StateFlow
13	import kotlinx.coroutines.launch
14	import timber.log.Timber
15	
16	class RiderListViewModel(private val title: String)
17	: ViewModel() {
18	
19	private val _riderList =
20	MutableStateFlow<List<KamenRider>>(emptyList())
21	val riderList: StateFlow<List<KamenRider>> =
22	_riderList
23	
24	private val _onClickRider =
25	MutableStateFlow<KamenRider?>(null)
26	val onClickRider: StateFlow<KamenRider?> =
27	_onClickRider
28	
29	init {
30	loadKamenRider()
31	}
32	
33	private fun loadKamenRider() {
34	viewModelScope.launch {
35	

36	val allRiders =
37	KamenRiderRepository.getKamenRiderList()
38	_riderList.value = allRiders
39	Timber.tag("ListViewModel").i("Data
40	item berhasil dimuat ke dalam list:
41	\${allRiders.size} item")
42	}
43	}
44	
45	fun onRiderClick(rider: KamenRider) {
46	_onClickRider.value = rider
47	}
48	
49	fun clearRiderClick() {
50	_onClickRider.value = null
51	}
	}
	class RiderListViewModelFactory(private val title: String) : ViewModelProvider.Factory {
	override fun <T : ViewModel> create(modelClass: Class<T>): T {
	if
	(modelClass.isAssignableFrom(RiderListViewModel::class.java)) {
	@Suppress("UNCHECKED_CAST")
	return RiderListViewModel(title) as T
	}
	throw IllegalArgumentException("Unknown ViewModel class")

	<pre> } } </pre>
--	------------------------------

DetailViewModel.kt

Tabel 30. Source Code Jetpack Compose ViewModel.kt Soal 1

1	package com.example.scrollablecompose.viewmodel
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider
5	import androidx.lifecycle.viewModelScope
6	import
7	com.example.scrollablecompose.model.KamenRider
8	import
9	com.example.scrollablecompose.model.KamenRiderRepo
10	sitory
11	import kotlinx.coroutines.flow.MutableStateFlow
12	import kotlinx.coroutines.flow.StateFlow
13	import kotlinx.coroutines.launch
14	import timber.log.Timber
15	
16	class DetailViewModel(private val id: Int) :
17	ViewModel() {
18	
19	private val _rider =
20	MutableStateFlow<KamenRider?>(null)
21	val rider: StateFlow<KamenRider?> get() = _rider
22	
23	init {
24	loadRiderById()
25	}

26	
27	private fun loadRiderById() {
28	viewModelScope.launch {
29	val selected =
30	KamenRiderRepository.getKamenRiderList().find {
31	it.id == id }
32	if (selected != null) {
33	
34	Timber.tag("DetailViewModel").i("Navigasi ke
35	DetailScreen dengan data: \${selected.name}, tahun:
36	\${selected.year}")
37	} else {
38	
39	Timber.tag("DetailViewModel").w("Rider dengan ID
40	\$id tidak ditemukan")
41	}
42	_rider.value = selected
43	}
44	}
	}
	 class DetailViewModelFactory(private val id: Int) : ViewModelProvider.Factory { override fun <T : ViewModel> create(modelClass: Class<T>): T { if (modelClass.isAssignableFrom(DetailViewModel::clas s.java)) { @Suppress("UNCHECKED_CAST") return DetailViewModel(id) as T

	<pre> } throw IllegalArgumentException("Unknown ViewModel class") } } </pre>
--	--

TimberApp.kt

Tabel 31. Source Code Jetpack Compose TimberApp.kt Soal 1

1	package com.example.scrollablecompose
2	
3	import android.app.Application
4	import timber.log.Timber
5	
6	class TimberApp : Application() {
7	override fun onCreate() {
8	super.onCreate()
9	
10	if (BuildConfig.DEBUG) {
11	Timber.plant(Timber.DebugTree())
12	}
13	}
14	}

2) Versi XML

MainAdapter.kt

Tabel 32. Source Code XML MainAdapter.kt

1	package com.example.scrollablexml.adapter
2	
3	import android.annotation.SuppressLint

4	import android.view.LayoutInflater
5	import android.view.ViewGroup
6	import androidx.recyclerview.widget.RecyclerView
7	import
8	com.example.scrollablexml.databinding.AdapterMainB
9	inding
10	import com.example.scrollablexml.model.KamenRider
11	import timber.log.Timber
12	import java.util.Locale
13	
14	class MainAdapter(
15	initialList: List<KamenRider>,
16	private val onImdbClick: (String) -> Unit,
17	private val onDetailClick: (KamenRider) -> Unit
18) : RecyclerView.Adapter<MainAdapter.ViewHolder>()
19	{
20	
21	private val riderList =
22	initialList.toMutableList()
23	
24	class ViewHolder(private val binding:
25	AdapterMainBinding) :
26	RecyclerView.ViewHolder(binding.root) {
27	fun bind(
28	rider: KamenRider,
29	onImdbClick: (String) -> Unit,
30	onDetailClick: (KamenRider) -> Unit
31) {
32	
33	

34	
35	binding.riderImage.setImageResource(rider.posterRe
36	s)
37	binding.riderName.text = rider.name
38	binding.riderYear.text =
39	String.format(Locale.getDefault(), "%d",
40	rider.year)
41	binding.descBody.text =
42	rider.description
43	binding.imdbBtn.setOnClickListener {
44	
45	Timber.tag("MainAdapter").i("Tombol Eksplisit
46	Intent ditekan untuk: \${rider.name}, URL:
47	\${rider.imdbUrl}")
48	onImdbClick(rider.imdbUrl)
49	}
50	binding.detailBtn.setOnClickListener {
51	
52	Timber.tag("MainAdapter").i("Tombol Detail ditekan
53	untuk: \${rider.name}, ID: \${rider.id}")
54	onDetailClick(rider)
55	}
56	}
57	}
58	
59	override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
	val binding =
	AdapterMainBinding.inflate(LayoutInflater.from(par
	ent.context), parent, false)

	<pre> return ViewHolder(binding) } override fun getItemCount(): Int = riderList.size override fun onBindViewHolder(holder: ViewHolder, position: Int) { holder.bind(riderList[position], onImdbClick, onDetailClick) } @SuppressWarnings("NotifyDataSetChanged") fun updateData(newList: List<KamenRider>) { riderList.clear() riderList.addAll(newList) notifyDataSetChanged() } } </pre>
--	---

ListFragment.kt

Tabel 33. Source Code XML ListFragment.kt

1	package com.example.scrollablexml.fragment
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.navigation.fragment.findNavController
14	import
15	com.example.scrollablexml.adapter.MainAdapter
16	import
17	com.example.scrollablexml.databinding.FragmentList
18	Binding
19	import
20	com.example.scrollablexml.viewmodel.RiderListViewM
21	odel
22	import
23	com.example.scrollablexml.viewmodel.RiderListViewM
24	odelFactory
25	import kotlinx.coroutines.flow.collectLatest
26	import kotlinx.coroutines.launch
27	
28	class ListFragment : Fragment() {
29	
30	private var _binding: FragmentListBinding? =
31	null
32	private val binding get() = _binding!!
33	
34	private val viewModel: RiderListViewModel by
35	viewModels {
36	RiderListViewModelFactory("Kamen Rider
37	List")
38	}

39	
40	private lateinit var mainAdapter: MainAdapter
41	
42	override fun onCreateView(
43	inflater: LayoutInflater, container:
44	ViewGroup?,
45	savedInstanceState: Bundle?
46): View {
47	_binding =
48	FragmentListBinding.inflate(inflater, container,
49	false)
50	
51	mainAdapter = MainAdapter(
52	initialList = emptyList(),
53	onImdbClick = { url ->
54	val intent =
55	Intent(Intent.ACTION_VIEW, Uri.parse(url)).apply {
56	
57	setPackage("com.android.chrome")
58	}
59	startActivity(intent)
60	},
61	onDetailClick = { rider ->
62	viewModel.onRiderClick(rider)
63	}
64)
65	
66	binding.recyclerView.adapter = mainAdapter
67	
68	observeViewModel()

69	
70	return binding.root
71	}
72	
73	private fun observeViewModel() {
74	viewLifecycleOwner.lifecycleScope.launch {
75	viewModel.riderList.collectLatest {
76	list ->
77	mainAdapter.updateData(list)
78	}
79	}
80	
	viewLifecycleOwner.lifecycleScope.launch {
	viewModel.onClickRider.collectLatest {
	rider ->
	rider?.let {
	val action =
	ListFragmentDirections.actionListFragmentToDetailF
	ragment(it.id)
	findNavController().navigate(action)
	viewModel.clearRiderClick()
	}
	}
	}
	}
	override fun onDestroyView() {
	super.onDestroyView()
	_binding = null

	}
	}

DetailFragment.kt

Tabel 34. Source Code XML DetailFragment.kt

1	package com.example.scrollablexml.fragment
2	
3	import android.os.Bundle
4	import android.view.LayoutInflater
5	import android.view.View
6	import android.view.ViewGroup
7	import androidx.fragment.app.Fragment
8	import androidx.lifecycle.ViewModelProvider
9	import androidx.lifecycle.LifecycleScope
10	import androidx.navigation.fragment.navArgs
11	import com.example.scrollablexml.R
12	import
13	com.example.scrollablexml.databinding.FragmentDeta
14	ilBinding
15	import
16	com.example.scrollablexml.viewmodel.DetailViewMode
17	l
18	import
19	com.example.scrollablexml.viewmodel.DetailViewMode
20	lFactory
21	import kotlinx.coroutines.flow.collectLatest
22	import kotlinx.coroutines.launch
23	
24	class DetailFragment : Fragment() {
25	

26	private var _binding: FragmentDetailBinding? =
27	null
28	private val binding get() = _binding!!
29	
30	private val args: DetailFragmentArgs by
31	navArgs()
32	
33	private val viewModel: DetailViewModel by lazy
34	{
35	val factory =
36	DetailViewModelFactory(args.riderId)
37	ViewModelProvider(this,
38	factory)[DetailViewModel::class.java]
39	}
40	
41	override fun onCreateView(
42	inflater: LayoutInflater, container:
43	ViewGroup?,
44	savedInstanceState: Bundle?
45): View {
46	_binding =
47	FragmentDetailBinding.inflate(inflater, container,
48	false)
49	
50	
51	binding.topAppBar.setNavigationOnClickListener {
52	
53	requireActivity().onBackPressedDispatcher.onBackPr
54	essed()
55	}

56	
57	observeRider()
58	
59	return binding.root
60	}
61	
62	private fun observeRider() {
63	viewLifecycleOwner.lifecycleScope.launch {
	viewModel.rider.collectLatest { rider -
>	
	rider?.let {
	binding.riderImage.setImageResource(it.posterRes)
	binding.riderName.text =
	it.name
	binding.riderYear.text =
	getString(R.string.detail_year, it.year)
	binding.descBody.text =
	it.description
	}
	}
	}
	override fun onDestroyView() {
	super.onDestroyView()
	_binding = null
	}
	}

RiderListViewModel.kt

Tabel 35. Source Code XML RiderListViewModel.kt

1	package com.example.scrollablexml.viewmodel
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider
5	import androidx.lifecycle.viewModelScope
6	import com.example.scrollablexml.model.KamenRider
7	import
8	com.example.scrollablexml.model.KamenRiderRepository
9	import kotlinx.coroutines.flow.MutableStateFlow
10	import kotlinx.coroutines.flow.StateFlow
11	import kotlinx.coroutines.launch
12	import timber.log.Timber
13	
14	class RiderListViewModel(private val title: String) :
15	ViewModel() {
16	
17	private val _riderList =
18	MutableStateFlow<List<KamenRider>>(emptyList())
19	val riderList: StateFlow<List<KamenRider>> =
20	_riderList
21	
22	private val _onClickRider =
23	MutableStateFlow<KamenRider?>(null)
24	val onClickRider: StateFlow<KamenRider?> =
25	_onClickRider
26	
27	init {
28	loadKamenRider()

```

29     }
30
31     private fun loadKamenRider() {
32         viewModelScope.launch {
33             val allRiders =
34 KamenRiderRepository.getKamenRiderList()
35             _riderList.value = allRiders
36             Timber.tag("ListViewModel").i("Data item
37 berhasil dimuat ke dalam list: ${allRiders.size}
38 item")
39         }
40     }
41
42     fun onRiderClick(rider: KamenRider) {
43         _onClickRider.value = rider
44     }
45
46     fun clearRiderClick() {
47         _onClickRider.value = null
48     }
49 }
50
51 class RiderListViewModelFactory(private val title:
String) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass:
Class<T>): T {
        if
        (modelClass.isAssignableFrom(RiderListViewModel::cla
ss.java)) {
            @Suppress("UNCHECKED_CAST")

```

	<pre> return RiderListViewModel(title) as T } throw IllegalArgumentException("Unknown ViewModel class") } } </pre>
--	--

DetailViewModel.kt

Tabel 36. Source Code XML DetailViewModel.kt

1	package com.example.scrollablexml.viewmodel
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider
5	import androidx.lifecycle.viewModelScope
6	import com.example.scrollablexml.model.KamenRider
7	import
8	com.example.scrollablexml.model.KamenRiderRepository
9	import kotlinx.coroutines.flow.MutableStateFlow
10	import kotlinx.coroutines.flow.StateFlow
11	import kotlinx.coroutines.launch
12	import timber.log.Timber
13	
14	class DetailViewModel(private val id: Int) :
15	ViewModel() {
16	
17	private val _rider =
18	MutableStateFlow<KamenRider?>(null)
19	val rider: StateFlow<KamenRider?> get() = _rider
20	
21	init {

```

22         loadRiderById()
23     }
24
25     private fun loadRiderById() {
26         viewModelScope.launch {
27             val selected =
28             KamenRiderRepository.getKamenRiderList().find {
29                 it.id == id }
30             if (selected != null) {
31
32                 Timber.tag("DetailViewModel").i("Navigasi ke
33                 DetailScreen dengan data: ${selected.name}, tahun:
34                 ${selected.year}")
35             } else {
36
37                 Timber.tag("DetailViewModel").w("Rider dengan ID $id
38                 tidak ditemukan")
39             }
40             _rider.value = selected
41         }
42     }
43 }
44
45 class DetailViewModelFactory(private val id: Int) :
46     ViewModelProvider.Factory {
47     override fun <T : ViewModel> create(modelClass:
48     Class<T>): T {
49         if
50         (modelClass.isAssignableFrom(DetailViewModel::class.
51         java)) {

```

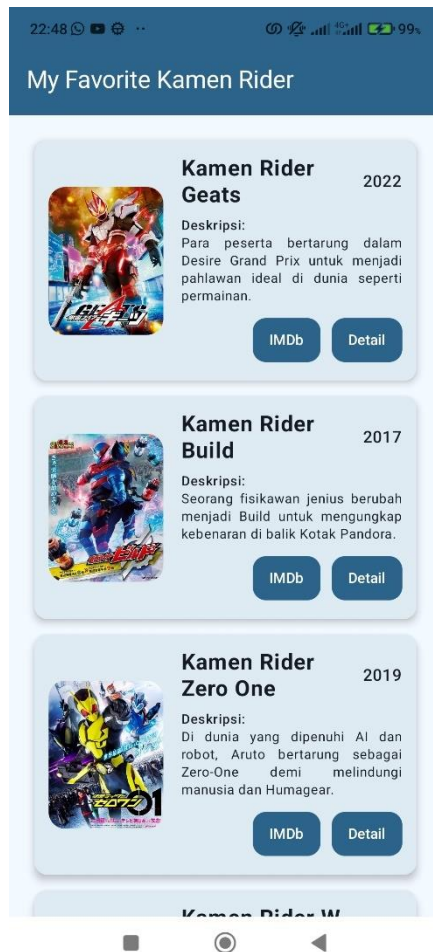
	<pre> @Suppress("UNCHECKED_CAST") return DetailViewModel(id) as T } throw IllegalArgumentException("Unknown ViewModel class") } } </pre>
--	--

TimberApp.kt

Tabel 37. Source Code XML TimberApp.kt

1	package com.example.scrollablexml
2	
3	import android.app.Application
4	import timber.log.Timber
5	
6	class TimberApp : Application() {
7	override fun onCreate() {
8	super.onCreate()
9	
10	if (BuildConfig.DEBUG) {
11	Timber.plant(Timber.DebugTree())
12	}
13	}
14	}

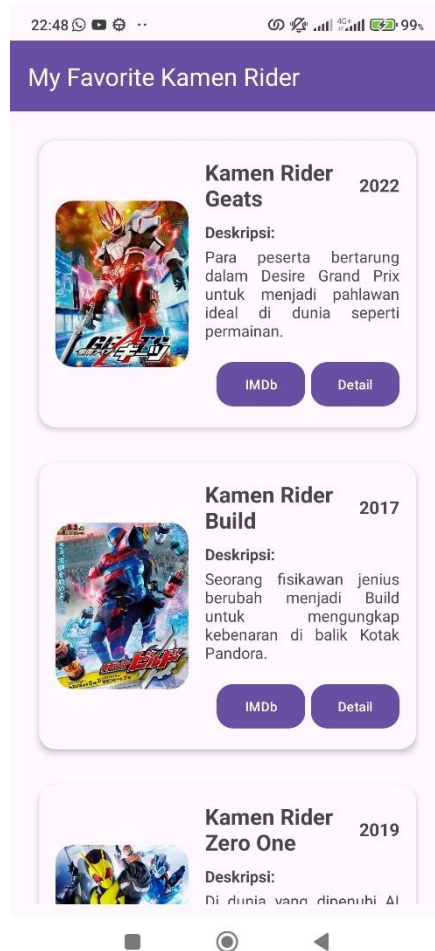
B. Output Program



Gambar 18. Screenshot List Screen Soal 1 Versi Jetpack Compose



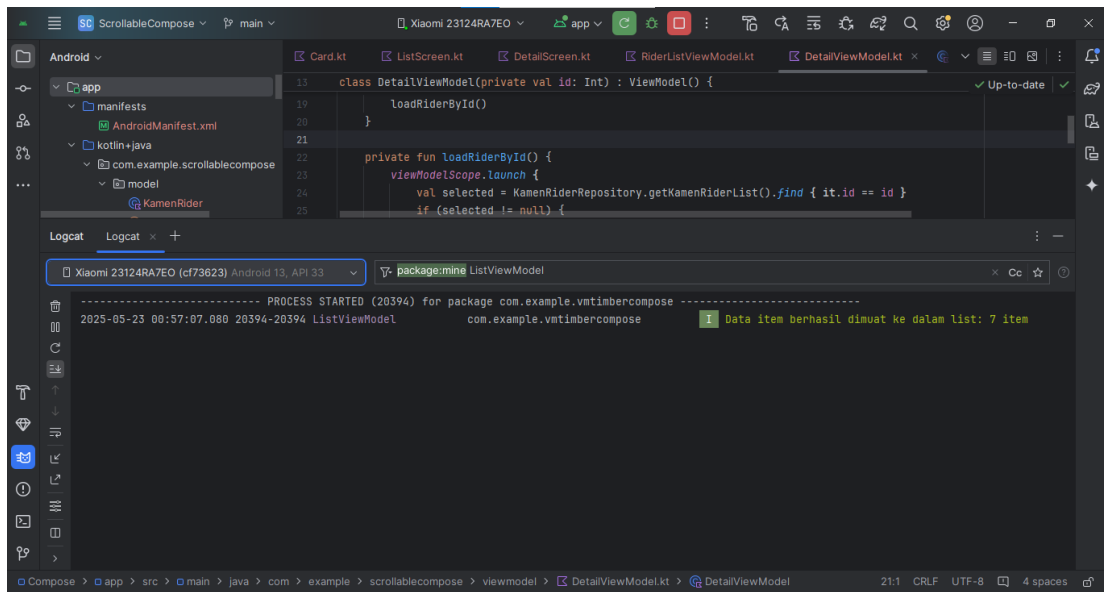
Gambar 19. Screenshot Detail Screen Soal 1 Versi Jetpack Compose



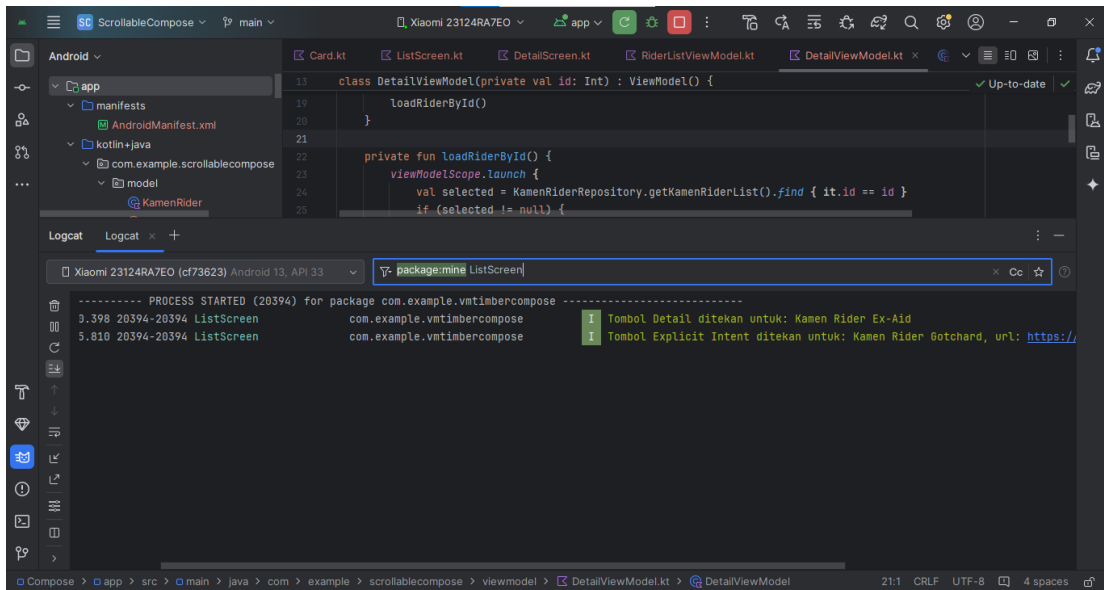
Gambar 20. Screenshot List Screen Soal 1 Versi XML



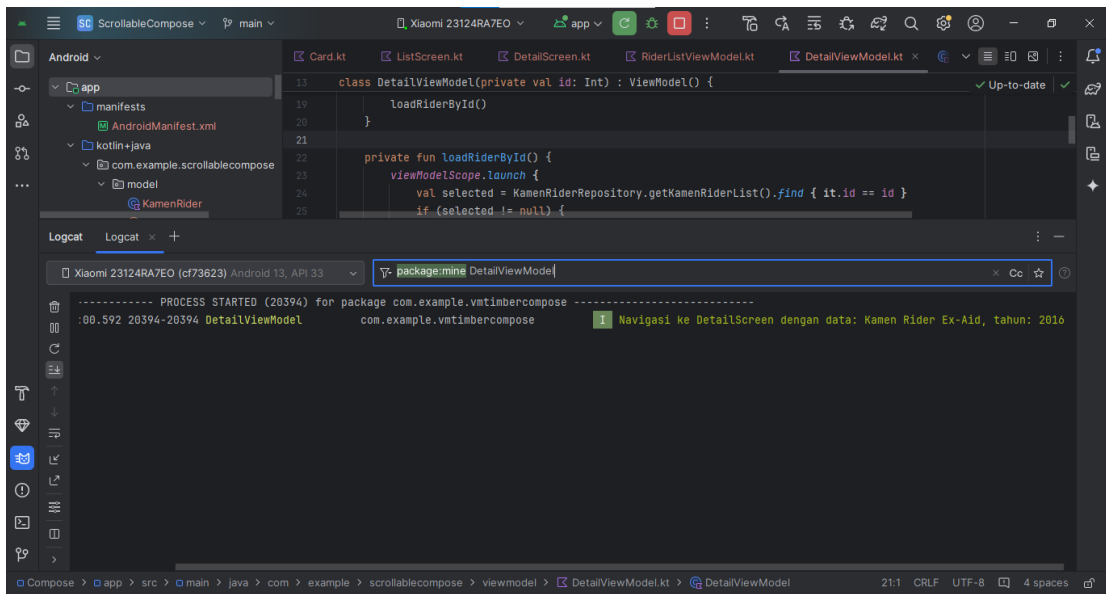
Gambar 21. Screenshot Detail Screen Soal 1 Versi XML



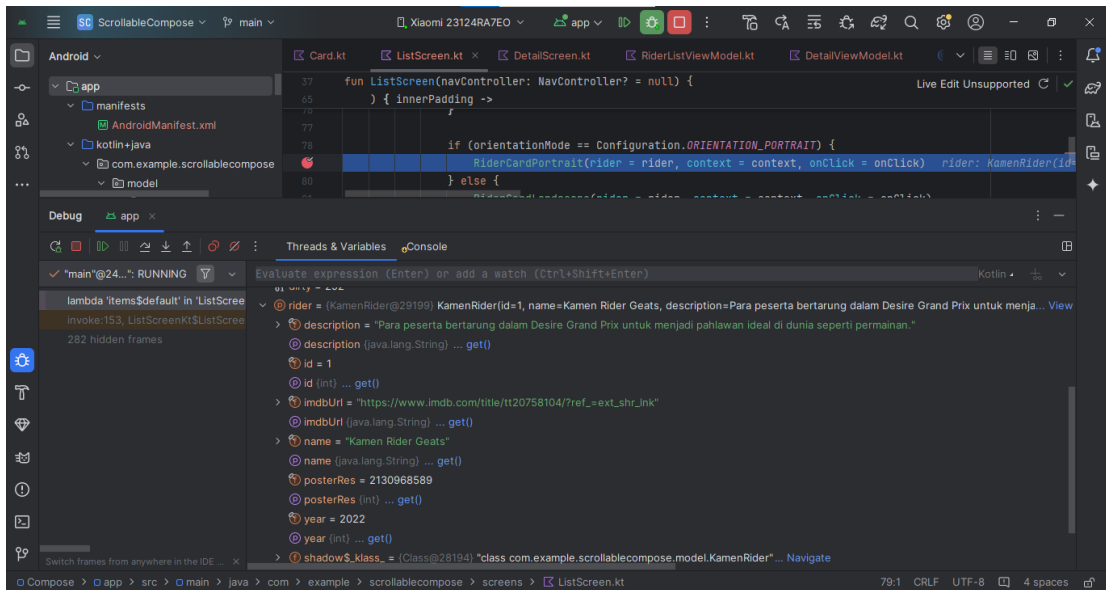
Gambar 22. Screenshot Log Saat Data Item Masuk Ke Dalam List Versi Compose



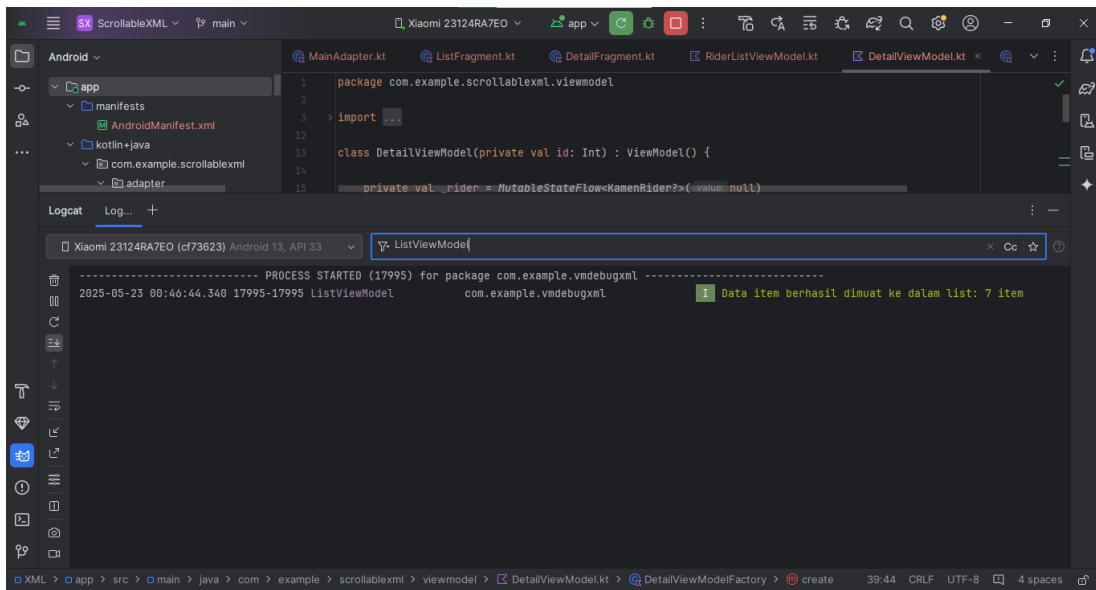
Gambar 23. Screenshot Log Saat Tombol Detail Dan Tombol Explicit Intent Ditekan Versi Compose



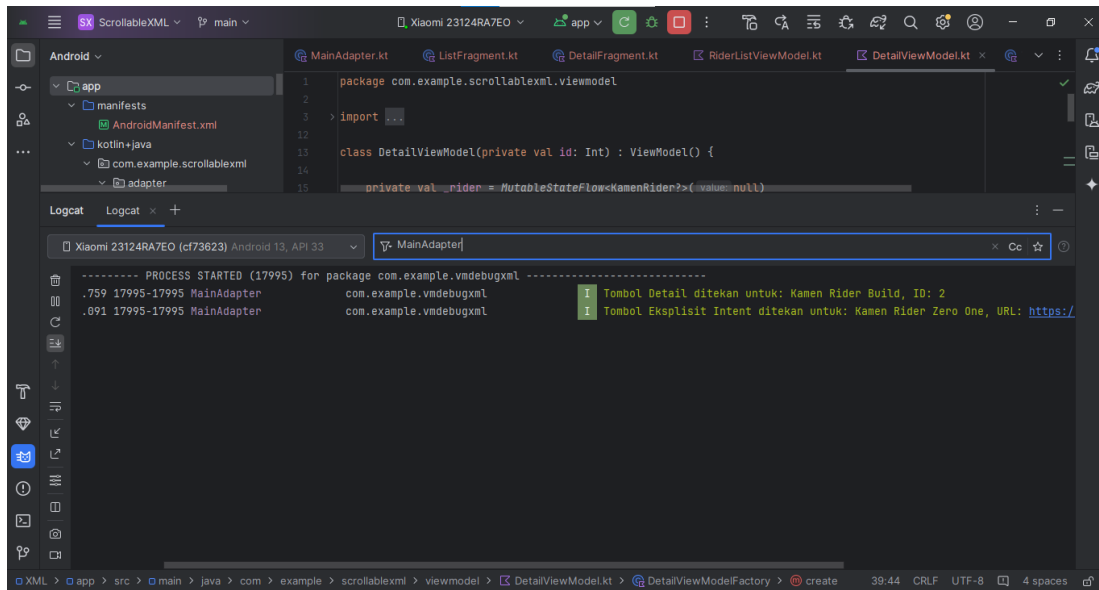
Gambar 24. Screenshot Log Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Versi Compose



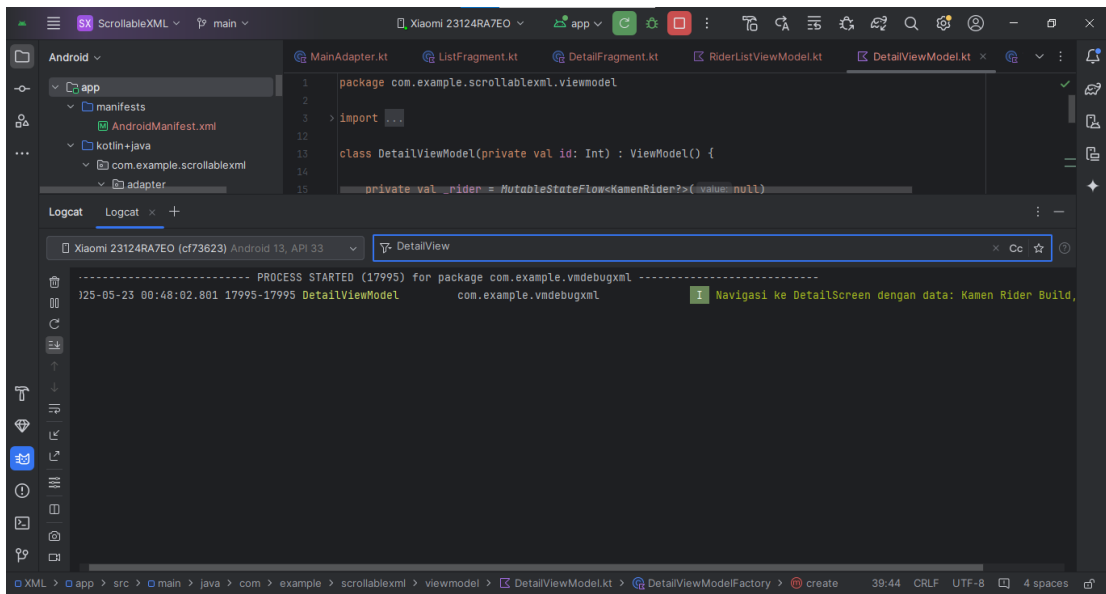
Gambar 25. Screenshot Debugging dengan Tool Debugger di Android Studio Versi Compose



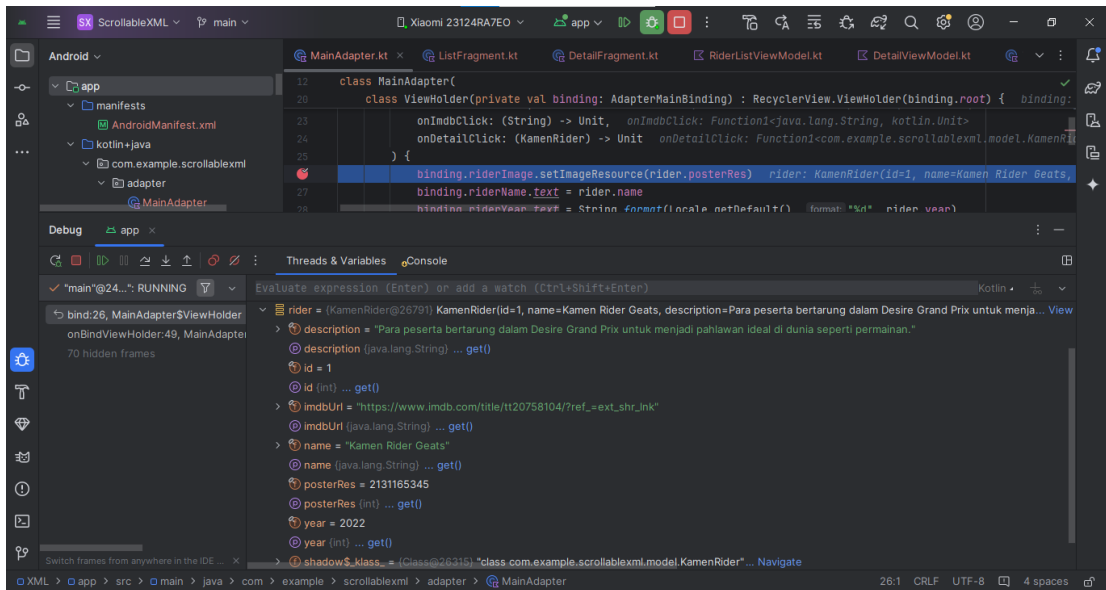
Gambar 26. Screenshot Log Saat Data Item Masuk Ke Dalam List Versi XML



Gambar 27. Screenshot Log Saat Tombol Detail Dan Tombol Explicit Intent Ditekan Versi XML



Gambar 28. Screenshot Log Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Versi XML



Gambar 29. Screenshot Debugging dengan Tool Debugger di Android Studio Versi XML

C. Pembahasan

1) Versi Jetpack Compose

Card.kt

Pada `Card.kt`, ditampilkan dua komponen UI utama dalam bentuk fungsi `Composable`, yaitu `RiderCardPortrait` dan `RiderCardLandscape`, yang masing-masing menampilkan kartu informasi Kamen Rider dalam orientasi potret dan lanskap. Kedua fungsi menerima objek `KamenRider`, `Context`, dan `onClick` lambda sebagai parameter untuk menangani aksi pengguna. Di dalamnya, kartu dibuat menggunakan `Card` dari `Material 3`, dengan desain responsif, bayangan (`elevation`), serta gaya warna yang konsisten dengan tema aplikasi. Gambar poster ditampilkan dengan `Image`, menggunakan `painterResource` untuk memuat resource lokal, dipadukan dengan `RoundedCornerShape` dan `ContentScale.FillBounds` agar tampilan tetap proporsional.

Teks judul (`name`), tahun (`year`), dan deskripsi ditampilkan secara terstruktur dengan `Text`, serta disesuaikan ukurannya tergantung orientasi. Dua tombol berada di bagian bawah: tombol “IMDB” menggunakan `explicit intent` yang diarahkan ke aplikasi `Chrome` dan membuka URL dari properti `imdbUrl`, sedangkan tombol “Detail” menjalankan `callback onClick`, biasanya digunakan untuk navigasi ke layar detail. Kedua tombol dilengkapi dengan `Timber log` yang mencatat saat tombol ditekan, membantu debugging dan pelacakan interaksi pengguna.

ListScreen.kt

Pada `ListScreen.kt`, ditampilkan antarmuka utama yang menampilkan daftar Kamen Rider menggunakan `LazyColumn` dalam arsitektur `Jetpack Compose`. Fungsi `ListScreen` dideklarasikan sebagai `@Composable` dan memanfaatkan `Scaffold` untuk menyusun tata letak dengan `TopAppBar` yang menampilkan judul aplikasi. `ViewModel` diinisialisasi menggunakan `RiderListViewModelFactory`, di mana datanya dikumpulkan secara reaktif melalui `collectAsState()` dari dua `StateFlow`: `riderList` untuk daftar Kamen Rider, dan `onClickRider` untuk menangani interaksi pengguna.

Mode orientasi perangkat diambil dari `LocalConfiguration`, digunakan untuk menentukan apakah `RiderCardPortrait` atau `RiderCardLandscape` yang akan ditampilkan. Komponen `LaunchedEffect` memantau perubahan pada `clickedRider`, dan jika pengguna memilih salah satu Rider, akan dilakukan navigasi ke `DetailScreen` berdasarkan ID Rider melalui `NavController`, lalu klik akan di-reset menggunakan `clearRiderClick()`. Masing-masing item daftar memiliki tombol "Detail" yang ketika ditekan, akan memicu `onRiderClick()` pada `ViewModel`, sekaligus dicatat dalam log menggunakan `Timber`.

File ini juga menyertakan `@Preview` untuk menampilkan pratinjau layar dalam mode potret (`ListScreenPortPreview`) dengan tema `ScrollableComposeTheme`, memudahkan proses desain antarmuka di Android Studio. Potongan kode pratinjau untuk mode lanskap juga tersedia, namun masih dikomentari.

DetailScreen.kt

Pada `DetailScreen.kt`, ditampilkan layar detail dari objek `KamenRider` tertentu berdasarkan ID yang diterima sebagai parameter. Komponen utama `DetailScreen` dibangun dengan `Scaffold` dan menampilkan `TopAppBar` yang berisi tombol navigasi kembali dan judul. `ViewModel` diinisialisasi menggunakan `DetailViewModelFactory`, dan data rider diambil secara reaktif menggunakan `collectAsState()` dari flow yang disediakan oleh `ViewModel`. Jika data Rider tersedia, tampilan akan menyesuaikan berdasarkan orientasi layar—mode potret akan memanggil `DetailPortrait`, sedangkan mode lanskap menggunakan `DetailLandscape`.

Tampilan potret (`DetailPortrait`) menyusun gambar dan teks dalam kolom vertikal yang dapat digulir, sementara tampilan lanskap (`DetailLandscape`) menyusunnya dalam baris horizontal untuk memanfaatkan ruang lebar layar. Komponen `RiderImage` bertugas menampilkan poster Rider dengan ukuran besar dan sudut membulat. Sementara itu, `RiderTextInfo` menampilkan informasi teks seperti nama,

tahun, dan deskripsi Rider, dengan format penataan teks yang menyesuaikan orientasi (rata tengah untuk potret dan rata kiri untuk lanskap).

Selain itu, file ini menyediakan pratinjau layar dengan anotasi `@Preview`, memungkinkan pengembang melihat tampilan antarmuka dalam mode potret melalui `DetailScreenPortraitPreview`, sedangkan pratinjau lanskap tersedia namun masih dikomentari. File ini mencerminkan pemisahan logika dan tampilan dengan baik, serta mengutamakan responsivitas terhadap orientasi perangkat untuk pengalaman pengguna yang optimal.

RiderListViewModel.kt

File `RiderListViewModel.kt` berfungsi sebagai bagian dari arsitektur MVVM, menangani logika data untuk menampilkan daftar Kamen Rider di UI. `RiderListViewModel` merupakan subclass dari `ViewModel` yang bertanggung jawab untuk memuat data dari `KamenRiderRepository` secara asinkron menggunakan coroutine di dalam `viewModelScope`. Data daftar Rider disimpan dalam `_riderList`, sebuah `MutableStateFlow` yang terekspos ke UI sebagai `StateFlow` agar bisa diobservasi secara reaktif. Selain itu, `ViewModel` ini juga menyimpan informasi tentang Rider yang diklik pengguna melalui `_onClickRider`, yang juga terekspos sebagai `StateFlow`.

Ketika `RiderListViewModel` diinisialisasi, ia langsung memanggil fungsi `loadKamenRider()` untuk mengisi data dari repository. Fungsi ini mengambil semua data Rider dan mengisi `StateFlow` dengan daftar tersebut, serta mencatat jumlah item yang dimuat menggunakan Timber untuk keperluan logging dan debugging. Fungsi `onRiderClick()` digunakan untuk menetapkan Rider yang diklik, yang kemudian dapat digunakan di layar UI untuk navigasi ke layar detail. Setelah navigasi terjadi, `clearRiderClick()` dipanggil untuk mereset nilai klik agar tidak terjadi navigasi berulang.

Terakhir, `RiderListViewModelFactory` disediakan untuk membuat instance `RiderListViewModel` secara dinamis, khususnya ketika `ViewModel` memerlukan parameter tambahan (title). Kelas ini mengimplementasikan `ViewModelProvider.Factory` dan memeriksa apakah tipe `ViewModel` yang diminta cocok sebelum mengembalikan instance-nya. File ini mencerminkan praktik yang baik dalam mengelola data dan interaksi pengguna dalam aplikasi berbasis Jetpack Compose.

DetailViewModel.kt

File `DetailViewModel.kt` merupakan bagian penting dari arsitektur MVVM dalam aplikasi, yang berfungsi untuk menangani logika bisnis dan pengambilan data secara spesifik untuk satu entitas Kamen Rider berdasarkan id. `DetailViewModel` adalah subclass dari `ViewModel`, yang menerima parameter id saat inisialisasi. ID ini digunakan untuk mencari satu data Rider dari repository melalui fungsi `loadRiderById()`, yang dijalankan di dalam `viewModelScope` menggunakan coroutine agar proses asinkron tidak memblokir UI.

Data hasil pencarian Rider disimpan dalam properti `_rider`, sebuah `MutableStateFlow` yang terekspos sebagai `StateFlow` agar bisa diobservasi oleh komponen UI secara reaktif. Jika Rider ditemukan, data tersebut disimpan ke dalam `_rider` dan sebuah log informasi dicatat menggunakan Timber, menampilkan nama dan tahun Rider yang dipilih. Jika tidak ditemukan, pesan peringatan juga dicatat melalui Timber, memberikan visibilitas terhadap potensi kesalahan ID.

Untuk pembuatan instance `DetailViewModel` yang memerlukan parameter id, kelas `DetailViewModelFactory` disediakan. Factory ini mengimplementasikan `ViewModelProvider.Factory` dan memastikan bahwa hanya `DetailViewModel` yang bisa dibuat dengan parameter yang sesuai. File ini menunjukkan pendekatan yang terstruktur dan efisien dalam mengelola data entitas tunggal dan mendukung navigasi detail yang dinamis dalam aplikasi Android berbasis Jetpack Compose.

TimberApp.kt

File TimberApp.kt merupakan kelas turunan dari Application, yang digunakan untuk menginisialisasi konfigurasi global saat aplikasi pertama kali dijalankan. Di sini, tujuan utamanya adalah untuk mengaktifkan Timber, sebuah library logging yang lebih fleksibel dan powerful dibandingkan Log bawaan Android. Di dalam method onCreate(), terdapat pengecekan terhadap BuildConfig.DEBUG. Jika aplikasi berjalan dalam mode debug (misalnya saat development), maka Timber.plant(Timber.DebugTree()) akan dipanggil untuk menanamkan DebugTree, yaitu implementasi default Timber yang mencetak log ke Logcat. Dengan cara ini, developer dapat menggunakan Timber.tag("TAG").d("Pesan") atau log lainnya tanpa perlu menulis tag dan level log secara manual. Konfigurasi ini memastikan bahwa logging hanya aktif saat debugging dan tidak ikut ditanamkan ke build produksi (release), sehingga menjaga performa dan keamanan data saat rilis.

2) Versi XML

MainAdapter.kt

Pada MainAdapter.kt, didefinisikan sebuah adapter untuk RecyclerView yang bertugas menampilkan daftar objek KamenRider menggunakan ViewBinding dari layout AdapterMainBinding. Kelas MainAdapter menerima tiga parameter: daftar awal initialList, lambda onImdbClick yang dipicu ketika tombol IMDb ditekan, dan onDetailClick yang dipicu ketika tombol detail ditekan. Adapter menyimpan daftar KamenRider dalam riderList, yang bisa diperbarui melalui fungsi updateData().

Di dalam kelas ViewHolder, fungsi bind() akan mengatur setiap item UI berdasarkan data KamenRider yang diberikan, seperti menampilkan gambar, nama, tahun, dan deskripsi. Dua tombol diatur dengan listener: satu untuk membuka URL

IMDb menggunakan `onImdbClick`, dan satu lagi untuk menavigasi ke detail rider dengan `onDetailClick`. Setiap aksi tombol juga dicatat menggunakan Timber untuk tujuan debugging. Adapter ini mengimplementasikan metode standar `RecyclerView.Adapter`: `onCreateViewHolder`, `onBindViewHolder`, dan `getItemCount`, serta menambahkan metode `updateData()` yang memperbarui seluruh daftar dan memanggil `notifyDataSetChanged()` untuk me-refresh tampilan.

ListFragment.kt

Pada `ListFragment.kt`, didefinisikan sebuah fragment yang bertugas menampilkan daftar Kamen Rider menggunakan `RecyclerView` dan mengatur navigasi serta aksi pengguna. Fragment ini menggunakan `ViewBinding` melalui `FragmentListBinding` untuk mengakses elemen UI dengan aman dan efisien. `ViewModel` `RiderListViewModel` diinisialisasi menggunakan delegasi by `viewModels` dengan sebuah factory yang menyuplai judul list.

Di dalam `onCreateView()`, adapter `MainAdapter` dibuat dengan dua callback: satu untuk membuka link IMDb melalui Intent eksplisit menggunakan aplikasi Chrome, dan satu lagi untuk memicu navigasi ke layar detail dengan memanggil `viewModel.onRiderClick()`. Adapter kemudian dihubungkan ke `RecyclerView` melalui `binding.recyclerView.adapter`.

Fungsi `observeViewModel()` bertugas untuk mengamati dua aliran `StateFlow` dari `ViewModel`. Pertama, `riderList` dipantau agar daftar dalam adapter diperbarui saat data berubah. Kedua, `onClickRider` diamati untuk menangani klik pada item. Ketika rider dipilih, fragment akan menavigasi ke `DetailFragment` dengan membawa ID rider menggunakan `Safe Args`, dan kemudian memanggil `clearRiderClick()` untuk menghindari navigasi ulang.

Terakhir, pada `onDestroyView()`, binding di-set ke null untuk mencegah memory leak, sesuai praktik terbaik dalam penggunaan `ViewBinding` di Fragment.

DetailFragment.kt

Pada `DetailFragment.kt`, didefinisikan sebuah fragment yang bertugas menampilkan detail dari item Kamen Rider yang dipilih oleh pengguna. Fragment ini menggunakan `ViewBinding` melalui `FragmentDetailBinding` untuk mengakses komponen UI secara langsung dan aman. Fragment menerima argumen `riderId` menggunakan delegasi `navArgs()` dari `Navigation Component`, yang memungkinkan pengiriman data antar fragment dengan aman.

`ViewModel` `DetailViewModel` diinisialisasi secara lazy menggunakan `ViewModelProvider` dengan `DetailViewModelFactory`, yang menerima `riderId` sebagai parameter untuk memuat data rider yang sesuai dari repository.

Pada `onCreateView()`, binding diinisialisasi, dan `topAppBar` diberi listener untuk menangani navigasi kembali menggunakan `onBackPressedDispatcher`, yang memungkinkan pengguna kembali ke tampilan sebelumnya.

Fungsi `observeRider()` digunakan untuk mengamati aliran data dari rider (tipe `StateFlow`) di dalam `ViewModel`. Ketika data rider tersedia, UI akan diperbarui dengan gambar, nama, tahun, dan deskripsi dari Kamen Rider yang bersangkutan.

Terakhir, pada `onDestroyView()`, binding di-set ke null untuk mencegah terjadinya memory leak — sebuah praktik standar dalam siklus hidup Fragment ketika menggunakan `ViewBinding`.

RiderListViewModel.kt

File `RiderListViewModel.kt` berfungsi sebagai bagian dari arsitektur MVVM, menangani logika data untuk menampilkan daftar Kamen Rider di UI. `RiderListViewModel` merupakan subclass dari `ViewModel` yang bertanggung jawab untuk memuat data dari `KamenRiderRepository` secara asinkron

menggunakan coroutine di dalam viewModelScope. Data daftar Rider disimpan dalam `_riderList`, sebuah `MutableStateFlow` yang terekspos ke UI sebagai `StateFlow` agar bisa diobservasi secara reaktif. Selain itu, `ViewModel` ini juga menyimpan informasi tentang Rider yang diklik pengguna melalui `_onClickRider`, yang juga terekspos sebagai `StateFlow`.

Ketika `RiderListViewModel` diinisialisasi, ia langsung memanggil fungsi `loadKamenRider()` untuk mengisi data dari repository. Fungsi ini mengambil semua data Rider dan mengisi `StateFlow` dengan daftar tersebut, serta mencatat jumlah item yang dimuat menggunakan `Timber` untuk keperluan logging dan debugging. Fungsi `onRiderClick()` digunakan untuk menetapkan Rider yang diklik, yang kemudian dapat digunakan di layar UI untuk navigasi ke layar detail. Setelah navigasi terjadi, `clearRiderClick()` dipanggil untuk mereset nilai klik agar tidak terjadi navigasi berulang.

Terakhir, `RiderListViewModelFactory` disediakan untuk membuat instance `RiderListViewModel` secara dinamis, khususnya ketika `ViewModel` memerlukan parameter tambahan (`title`). Kelas ini mengimplementasikan `ViewModelProvider.Factory` dan memeriksa apakah tipe `ViewModel` yang diminta cocok sebelum mengembalikan instance-nya. File ini mencerminkan praktik yang baik dalam mengelola data dan interaksi pengguna dalam aplikasi berbasis Jetpack Compose.

DetailViewModel.kt

File `DetailViewModel.kt` merupakan bagian penting dari arsitektur MVVM dalam aplikasi, yang berfungsi untuk menangani logika bisnis dan pengambilan data secara spesifik untuk satu entitas Kamen Rider berdasarkan id. `DetailViewModel` adalah subclass dari `ViewModel`, yang menerima parameter id saat inisialisasi. ID ini digunakan untuk mencari satu data Rider dari repository melalui fungsi

loadRiderById(), yang dijalankan di dalam viewModelScope menggunakan coroutine agar proses asinkron tidak memblokir UI.

Data hasil pencarian Rider disimpan dalam properti `_rider`, sebuah MutableStateFlow yang terekspos sebagai StateFlow agar bisa diobservasi oleh komponen UI secara reaktif. Jika Rider ditemukan, data tersebut disimpan ke dalam `_rider` dan sebuah log informasi dicatat menggunakan Timber, menampilkan nama dan tahun Rider yang dipilih. Jika tidak ditemukan, pesan peringatan juga dicatat melalui Timber, memberikan visibilitas terhadap potensi kesalahan ID.

Untuk pembuatan instance DetailViewModel yang memerlukan parameter id, kelas DetailViewModelFactory disediakan. Factory ini mengimplementasikan ViewModelProvider.Factory dan memastikan bahwa hanya DetailViewModel yang bisa dibuat dengan parameter yang sesuai. File ini menunjukkan pendekatan yang terstruktur dan efisien dalam mengelola data entitas tunggal dan mendukung navigasi detail yang dinamis dalam aplikasi Android berbasis Jetpack Compose.

TimberApp.kt

File TimberApp.kt merupakan kelas turunan dari Application, yang digunakan untuk menginisialisasi konfigurasi global saat aplikasi pertama kali dijalankan. Di sini, tujuan utamanya adalah untuk mengaktifkan Timber, sebuah library logging yang lebih fleksibel dan powerful dibandingkan Log bawaan Android. Di dalam method onCreate(), terdapat pengecekan terhadap BuildConfig.DEBUG. Jika aplikasi berjalan dalam mode debug (misalnya saat development), maka Timber.plant(Timber.DebugTree()) akan dipanggil untuk menanamkan DebugTree, yaitu implementasi default Timber yang mencetak log ke Logcat. Dengan cara ini, developer dapat menggunakan Timber.tag("TAG").d("Pesan") atau log lainnya tanpa perlu menulis tag dan level log secara manual. Konfigurasi ini memastikan bahwa logging hanya aktif saat debugging dan tidak ikut

ditanamkan ke build produksi (release), sehingga menjaga performa dan keamanan data saat rilis.

3) Penjelasan Debugger

Debugger adalah alat bantu dalam IDE (Integrated Development Environment) seperti Android Studio yang digunakan untuk menjalankan aplikasi secara step-by-step guna menemukan dan memperbaiki bug atau kesalahan logika pada kode. Dengan debugger, pengembang dapat menghentikan eksekusi program pada titik tertentu yang disebut breakpoint, kemudian memeriksa nilai variabel, kondisi objek, dan alur logika secara langsung saat program berjalan.

Untuk menggunakan debugger, pertama-tama pengguna harus menempatkan breakpoint pada baris kode yang ingin dianalisis, biasanya dengan mengklik di sisi kiri editor kode. Setelah itu, aplikasi dijalankan dalam debug mode, dan ketika eksekusi mencapai breakpoint, aplikasi akan berhenti sementara sehingga pengguna bisa memeriksa status aplikasi melalui jendela debug.

Fitur Step Into digunakan untuk masuk ke dalam fungsi yang sedang dipanggil agar dapat melihat bagaimana fungsi tersebut bekerja baris per baris. Step Over digunakan untuk melangkahi eksekusi fungsi tanpa masuk ke dalamnya — berguna jika pengguna yakin bahwa fungsi tersebut berjalan dengan benar dan ingin melanjutkan ke baris berikutnya di luar fungsi. Sementara itu, Step Out digunakan untuk keluar dari fungsi saat ini dan kembali ke fungsi pemanggil, berguna ketika pengguna telah selesai menelusuri bagian dalam suatu fungsi dan ingin kembali ke konteks sebelumnya. Ketiga fitur ini sangat penting untuk memahami alur eksekusi kode secara mendalam dan menyeluruh.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya!

Jawab:

Kelas Application dalam arsitektur aplikasi Android merupakan komponen global yang pertama kali diinisialisasi saat aplikasi dijalankan, sebelum Activity, Service, atau BroadcastReceiver apa pun. Kelas ini berfungsi sebagai titik masuk utama dan wadah untuk mengelola konfigurasi atau inisialisasi global yang dibutuhkan sepanjang siklus hidup aplikasi.

Umumnya, Application digunakan untuk menginisialisasi dependency injection (seperti Hilt atau Dagger), konfigurasi database lokal (seperti Room), pengaturan library logging (seperti Timber), integrasi Firebase, atau crash reporting tools seperti Crashlytics. Untuk menggunakannya, pengembang membuat subclass dari Application, menambahkan logika pada metode onCreate(), dan mendeklarasikannya di AndroidManifest.xml.

Meskipun memiliki akses global, Application tidak dianjurkan untuk menyimpan state UI atau Context dari komponen seperti Activity atau Fragment, karena berisiko menyebabkan memory leak. Sebaliknya, penyimpanan state sebaiknya dikelola melalui ViewModel atau Repository sesuai prinsip arsitektur MVVM atau Clean Architecture.

Dengan peran strategisnya, kelas Application menjadi fondasi penting untuk memastikan konsistensi, efisiensi, dan skalabilitas aplikasi Android.

MODUL 5 : CONNECT TO THE INTERNET

SOAL 1

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

- a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
- b. Gunakan KotlinX Serialization sebagai library JSON.
- c. Gunakan library seperti Coil atau Glide untuk image loading.
- d. API yang digunakan pada modul ini adalah The Movie Database (TMDB) API yang menampilkan data film.

Berikut link dokumentasi API: <https://developer.themoviedb.org/docs/getting-started>

- e. Implementasikan konsep data persistence (aplikasi menyimpan data walau pengguna keluar dari aplikasi) dengan SharedPreferences untuk menyimpan data ringan (seperti pengaturan aplikasi) dan Room untuk data relasional.
- f. Gunakan caching strategy pada Room. Dibebaskan untuk memilih caching strategy yang sesuai, dan sertakan penjelasan kenapa menggunakan caching strategy tersebut.
- g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

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

A. Source Code

AppDatabase.kt

Tabel 38. Source Code AppDatabase.kt

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

27	instance
28	}
29	}
30	}
	}

MovieDao.kt

Tabel 39. Source Code MovieDao.kt

1	package com.pemrogramanmobile.apicompose.data.local
2	
3	import androidx.room.Dao
4	import androidx.room.Insert
5	import androidx.room.OnConflictStrategy
6	import androidx.room.Query
7	import kotlinx.coroutines.flow.Flow
8	
9	@Dao
10	interface MovieDao {
11	@Insert(onConflict =
12	OnConflictStrategy.REPLACE)
	suspend fun insertMovies(movies:
13	List<MovieEntity>)
14	
	@Query("SELECT * FROM movies ORDER BY
15	voteAverage DESC")
16	fun getAllMovies(): Flow<List<MovieEntity>>
17	
18	@Query("SELECT * FROM movies WHERE id =
	:movieId")

19	fun	getMovieById(movieId: Int):	
20	Flow<MovieEntity?>		
21			
22	@Query("DELETE FROM movies")		
23	suspend fun deleteAllMovies()		
24			
25	@Query("SELECT COUNT(*) FROM movies")		
26	suspend fun getMovieCount(): Int		
27	@Query("SELECT lastRefreshed FROM movies ORDER		
28	BY lastRefreshed DESC LIMIT 1")		
	suspend fun getLastRefreshTimestamp(): Long?		
	}		

MovieEntity.kt

Tabel 40. Source Code MovieEntity.kt

1	package com.pemrogramanmobile.apicompose.data.local
2	
3	import androidx.room.Entity
4	import androidx.room.PrimaryKey
5	
6	@Entity(tableName = "movies")
7	data class MovieEntity(
8	@PrimaryKey val id: Int,
9	val title: String,
10	val overview: String,
11	val posterPath: String?,
12	val releaseDate: String?,
13	val voteAverage: Double,

14	val lastRefreshed: Long =
	System.currentTimeMillis(),
15	val homepage: String? = null
16)

MovieApiModel.kt

Tabel 41. Source Code MovieApiModel.kt

1	package
2	com.pemrogramanmobile.apicompose.data.remote
3	
4	import kotlinx.serialization.SerialName
5	import kotlinx.serialization.Serializable
6	
7	@Serializable
8	data class MovieListResponse(
9	@SerialName("page") val page: Int,
	@SerialName("results") val results:
10	List<MovieApiModel>,
	@SerialName("total_pages") val totalPages: Int,
11	@SerialName("total_results") val totalResults:
	Int
12)
13	
14	@Serializable
15	data class MovieApiModel(
16	@SerialName("id") val id: Int,
17	@SerialName("title") val title: String,
18	@SerialName("overview") val overview: String,
19	

20	<pre> @SerializedName("poster_path") val posterPath: String?, @SerializedName("release_date") val releaseDate: 21 String?, @SerializedName("vote_average") val voteAverage: 22 Double, @SerializedName("homepage") val homepage: String? = 23 null) </pre>
----	---

RetrofitClient.kt

Tabel 42. Source Code RetrofitClient.kt

1	package
2	com.pemrogramanmobile.apicompose.data.remote
3	
	import
	com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverterFactory
4	
5	import kotlinx.serialization.json.Json
6	import okhttp3.MediaType.Companion.toMediaType
7	import okhttp3.OkHttpClient
8	import okhttp3.logging.HttpLoggingInterceptor
9	import retrofit2.Retrofit
10	
11	object RetrofitClient {
	private const val BASE_URL =
12	"https://api.themoviedb.org/3/"
	private const val API_KEY =
13	"91a08b3c95e46f3b968e7dfdfc6c81b0"

14	
15	private val json = Json {
16	ignoreUnknownKeys = true
17	coerceInputValues = true
18	}
19	
20	private val loggingInterceptor =
21	HttpLoggingInterceptor().apply {
22	level = HttpLoggingInterceptor.Level.BODY
23	}
24	private val httpClient = OkHttpClient.Builder()
25	.addInterceptor(loggingInterceptor)
26	.addInterceptor { chain ->
27	val originalRequest = chain.request()
28	val originalHttpRequest =
29	originalRequest.url
30	val url = originalHttpRequest.newBuilder()
31	.addQueryParameter("api_key",
32	API_KEY)
33	.build()
34	val requestBuilder =
35	originalRequest.newBuilder().url(url)
36	val request = requestBuilder.build()
37	chain.proceed(request)
38	}
39	.build()
40	
41	val instance: TmdbApiService by lazy {
42	Retrofit.Builder()

	<code>.baseUrl (BASE_URL)</code>
40	<code>.client (httpClient)</code>
41	
42	<code>.addConverterFactory (json.asConverterFactory ("appl</code>
43	<code>ication/json".toMediaType ()))</code>
44	<code>.build()</code>
45	<code>.create (TmdbApiService::class.java)</code>
	<code>}</code>
	<code>}</code>

TmdbApiService.kt

Tabel 43. Source Code TmdbApiService.kt

1	<code>package</code>
2	<code>com.pemrogramanmobile.apicompose.data.remote</code>
3	
4	<code>import retrofit2.http.GET</code>
5	<code>import retrofit2.http.Path</code>
6	<code>import retrofit2.http.Query</code>
7	
8	<code>interface TmdbApiService {</code>
9	<code> @GET ("movie/popular")</code>
10	<code> suspend fun getPopularMovies (</code>
11	<code> @Query ("page") page: Int = 1</code>
12	<code>): MovieListResponse</code>
13	
14	<code> @GET ("movie/{movie_id}")</code>
15	<code> suspend fun getMovieDetails (</code>
16	<code> @Path ("movie_id") movieId: Int</code>
17	<code>): MovieApiModel</code>

	}
--	---

MovieRepositoryImpl.kt

Tabel 44. Source Code MovieRepositoryImpl.kt

1	package
	com.pemrogramanmobile.apicompose.data.repository
2	
3	import android.content.Context
4	import
	com.pemrogramanmobile.apicompose.data.local.MovieD
5	ao
	import
	com.pemrogramanmobile.apicompose.data.remote.TmdbA
6	piService
	import
7	com.pemrogramanmobile.apicompose.data.toDomainMovi
	e
	import
8	com.pemrogramanmobile.apicompose.data.toDomainMovi
	esFromEntities
	import
9	com.pemrogramanmobile.apicompose.data.toMovieEntit
	ies
10	import
	com.pemrogramanmobile.apicompose.data.toMovieEntit
11	y
	import
	com.pemrogramanmobile.apicompose.domain.model.Movi
12	e

	import
13	com.pemrogramanmobile.apicompose.domain.repository
14	.MovieRepository
15	import
16	com.pemrogramanmobile.apicompose.util.NetworkResult
17	t
18	import kotlinx.coroutines.flow.Flow
19	import kotlinx.coroutines.flow.firstOrNull
20	import kotlinx.coroutines.flow.flow
21	import kotlinx.coroutines.flow.map
22	import java.util.concurrent.TimeUnit
23	
24	class MovieRepositoryImpl(
	private val tmdbApiService: TmdbApiService,
25	private val movieDao: MovieDao,
26	private val context: Context
) : MovieRepository {
27	private val CACHE_EXPIRY_MS =
	TimeUnit.HOURS.toMillis(1)
28	
	override fun getPopularMovies():
	Flow<NetworkResult<List<Movie>>> = flow {
29	val lastRefreshTime =
30	movieDao.getLastRefreshTimestamp() ?: 0L
31	val isCacheStale =
32	(System.currentTimeMillis() - lastRefreshTime) >
33	CACHE_EXPIRY_MS
	val movieCount = movieDao.getMovieCount()
	val isCacheEmpty = movieCount == 0
34	

35	if (!isCacheEmpty && !isCacheStale) {
36	val cachedDataFlow: Flow<List<Movie>> =
	movieDao.getAllMovies().map { entities ->
37	entities.toDomainMoviesFromEntities()
	}
38	val cachedData =
39	cachedDataFlow.firstOrNull()
40	if (cachedData != null &&
41	cachedData.isNotEmpty()) {
42	
43	emit(NetworkResult.Success(cachedData))
44	}
	}
45	if (isCacheEmpty isCacheStale) {
46	try {
	val movieListResponse =
	tmdbApiService.getPopularMovies()
47	val currentTime =
48	System.currentTimeMillis()
49	val movieEntities =
50	movieListResponse.results.toMovieEntities(currentT
51	ime)
	movieDao.deleteAllMovies()
52	
53	movieDao.insertMovies(movieEntities)
54	

55	val freshDataFlow:
56	Flow<List<Movie>> = movieDao.getAllMovies().map {
	entities ->
57	
58	entities.toDomainMoviesFromEntities()
59	}
	val freshDataFromDb =
	freshDataFlow.firstOrNull()
60	
61	if (freshDataFromDb != null) {
62	
	emit(NetworkResult.Success(freshDataFromDb))
	} else {
63	emit(NetworkResult.Error("Gagal mengambil data dari
64	database setelah refresh.", null))
65	}
	} catch (e: Exception) {
66	val fallbackDataFlow:
	Flow<List<Movie>> = movieDao.getAllMovies().map {
	entities ->
67	
68	entities.toDomainMoviesFromEntities()
69	}
70	val currentCachedData =
71	fallbackDataFlow.firstOrNull()
	emit(NetworkResult.Error("Gagal
72	memuat data dari jaringan: \${e.message}",
	currentCachedData))
73	}

	<pre> } } override fun getMovieDetails(movieId: Int): Flow<NetworkResult<Movie>> = flow { val cachedMovieEntity = movieDao.getMovieById(movieId).firstOrNull() if (cachedMovieEntity != null) { emit(NetworkResult.Success(cachedMovieEntity.toDomainMovie())) } try { val movieApiModel = tmdbApiService.getMovieDetails(movieId) val movieEntity = movieApiModel.toMovieEntity(System.currentTimeMillis()) movieDao.insertMovies(listOf(movieEntity)) emit(NetworkResult.Success(movieApiModel.toDomainMovie())) } catch (e: Exception) { if (cachedMovieEntity == null) { emit(NetworkResult.Error("Gagal memuat detail film: \${e.message}", null)) </pre>
--	--

	<pre> } else { emit(NetworkResult.Error("Gagal menyegarkan detail film: \${e.message}", cachedMovieEntity.toDomainMovie())) } } } } </pre>
--	--

Mappers.kt

Tabel 45. Source Code Mappers.kt

1	package com.pemrogramanmobile.apicompose.data
2	
3	import
	com.pemrogramanmobile.apicompose.data.local.MovieE
	ntity
4	import
	com.pemrogramanmobile.apicompose.data.remote.Movie
	ApiModel
5	import
	com.pemrogramanmobile.apicompose.domain.model.Movi
6	e
7	
8	fun MovieApiModel.toDomainMovie(): Movie {
9	return Movie(
10	id = this.id,
11	title = this.title,
12	overview = this.overview,
13	posterPath = this.posterPath,

14	releaseDate = this.releaseDate,
15	voteAverage = this.voteAverage,
16	homepage = this.homepage
17)
18	}
19	
	fun MovieApiModel.toMovieEntity(lastRefreshedTime:
20	Long): MovieEntity {
21	return MovieEntity(
22	id = this.id,
23	title = this.title,
24	overview = this.overview,
25	posterPath = this.posterPath,
26	releaseDate = this.releaseDate,
27	voteAverage = this.voteAverage,
28	lastRefreshed = lastRefreshedTime,
29	homepage = this.homepage
30)
31	}
32	
33	fun MovieEntity.toDomainMovie(): Movie {
34	return Movie(
35	id = this.id,
36	title = this.title,
37	overview = this.overview,
38	posterPath = this.posterPath,
39	releaseDate = this.releaseDate,
40	voteAverage = this.voteAverage,
41	homepage = this.homepage
42)

43	}
44	
	fun List<MovieApiModel>.toDomainMovies():
45	List<Movie> {
46	return this.map { it.toDomainMovie() }
47	}
48	
	fun List<MovieEntity>.toDomainMoviesFromEntities():
49	List<Movie> {
50	return this.map { it.toDomainMovie() }
51	}
52	
	fun
	List<MovieApiModel>.toMovieEntities(lastRefreshedT
53	ime: Long): List<MovieEntity> {
	return this.map {
54	it.toMovieEntity(lastRefreshedTime) }
	}

Movie.kt

Tabel 46. Source Code Movie.kt

1	package
	com.pemrogramanmobile.apicompose.domain.model
2	
3	data class Movie(
4	val id: Int,
5	val title: String,
6	val overview: String,
7	val posterPath: String?,

8	val releaseDate: String?,
9	val voteAverage: Double,
10	val homepage: String? = null
11)

MovieRepository.kt

Tabel 47. Source Code MovieRepository.kt

1	package
	com.pemrogramanmobile.apicompose.domain.repository
2	
3	import
	com.pemrogramanmobile.apicompose.domain.model.Movie
4	import
	com.pemrogramanmobile.apicompose.util.NetworkResult
5	import kotlinx.coroutines.flow.Flow
6	
7	interface MovieRepository {
8	fun getPopularMovies() :
	Flow<NetworkResult<List<Movie>>>
9	fun getMovieDetails(movieId: Int) :
	Flow<NetworkResult<Movie>>
10	}

GetMovieDetailsUseCase.kt

Tabel 48. Source Code GetMovieDetailsUseCase.kt

1	package
	com.pemrogramanmobile.apicompose.domain.usecase
2	

3	import
	com.pemrogramanmobile.apicompose.domain.model.Movi
4	e
	import
	com.pemrogramanmobile.apicompose.domain.repository
5	.MovieRepository
	import
6	com.pemrogramanmobile.apicompose.util.NetworkResul
7	t
8	import kotlinx.coroutines.flow.Flow
9	class GetMovieDetailsUseCase(private val
	movieRepository: MovieRepository) {
10	operator fun invoke(movieId: Int):
	Flow<NetworkResult<Movie>> {
11	return
12	movieRepository.getMovieDetails(movieId)
	}
	}

GetPopularMoviesUseCase.kt

Tabel 49. Source Code GetPopularMoviesUseCase.kt

1	package
	com.pemrogramanmobile.apicompose.domain.usecase
2	
3	import
	com.pemrogramanmobile.apicompose.domain.model.Movi
4	e

5	import
	com.pemrogramanmobile.apicompose.domain.repository
6	.MovieRepository
7	import
8	com.pemrogramanmobile.apicompose.util.NetworkResult
	t
9	import kotlinx.coroutines.flow.Flow
10	class GetPopularMoviesUseCase(private val
11	movieRepository: MovieRepository) {
12	operator fun invoke():
	Flow<NetworkResult<List<Movie>>> {
	return movieRepository.getPopularMovies()
	}
	}

MovieDetailScreen.kt

Tabel 50. Source Code MovieDetailScreen.kt

1	package
	com.pemrogramanmobile.apicompose.ui.screen.moviedetail
2	
3	import android.content.ActivityNotFoundException
4	import android.content.Intent
5	import android.net.Uri
6	import android.widget.Toast
7	import
	androidx.compose.foundation.layout.Arrangement
8	import androidx.compose.foundation.layout.Box
9	import androidx.compose.foundation.layout.Column

10	import androidx.compose.foundation.layout.Row
11	import androidx.compose.foundation.layout.Spacer
12	import androidx.compose.foundation.layout.aspectRatio
13	import androidx.compose.foundation.layout.fillMaxSize
14	import androidx.compose.foundation.layout.fillMaxWidth
15	import androidx.compose.foundation.layout.height
16	import androidx.compose.foundation.layout.padding
17	import androidx.compose.foundation.layout.size
18	import androidx.compose.foundation.rememberScrollState
19	import androidx.compose.foundation.shape.RoundedCornerSha
20	pe
21	import androidx.compose.foundation.verticalScroll
22	import androidx.compose.material.icons.Icons import androidx.compose.material.icons.automirrored.fille
23	d.ArrowBack
24	import androidx.compose.material.icons.filled.Search
25	import androidx.compose.material3.Button
26	import androidx.compose.material3.ButtonDefaults
27	import androidx.compose.material3.CircularProgressIndicat
28	or
29	
30	

31	import
32	androidx.compose.material3.ExperimentalMaterial3Ap
33	i
34	import androidx.compose.material3.Icon
35	import androidx.compose.material3.IconButton
36	import androidx.compose.material3.MaterialTheme
37	import androidx.compose.material3.Scaffold
38	import androidx.compose.material3.SnackbarDuration
39	import androidx.compose.material3.SnackbarHost
40	import androidx.compose.material3.SnackbarHostState
41	import androidx.compose.material3.Text
42	import androidx.compose.material3.TopAppBar
43	import androidx.compose.material3.TopAppBarDefaults
44	import androidx.compose.runtime.Composable
45	import androidx.compose.runtime.LaunchedEffect
46	import androidx.compose.runtime.getValue
47	import androidx.compose.runtime.remember
48	import androidx.compose.ui.Alignment
49	import androidx.compose.ui.Modifier
50	import androidx.compose.ui.draw.clip
51	import androidx.compose.ui.layout.ContentScale
52	import androidx.compose.ui.platform.LocalContext
	import androidx.compose.ui.res.painterResource
	import androidx.compose.ui.res.stringResource
53	import androidx.compose.ui.text.font.FontWeight
	import androidx.compose.ui.text.style.TextAlign
54	import androidx.compose.ui.unit.dp
55	import
56	androidx.lifecycle.compose.collectAsStateWithLifec
57	ycle

58	import	
	androidx.lifecycle.viewmodel.compose.viewModel	
59	import androidx.navigation.NavController	
	import coil.compose.AsyncImage	
	import coil.request.ImageRequest	
60	import com.pemrogramanmobile.apicompose.R	
61	import	
62	com.pemrogramanmobile.apicompose.domain.model.Movi	
63	e	
64	import	
65	com.pemrogramanmobile.apicompose.ui.viewmodel.Movi	
	eDetailViewModel	
66		
67	@OptIn(ExperimentalMaterial3Api::class)	
	@Composable	
68	fun MovieDetailScreen(
	navController: NavController,	
69	viewModel: MovieDetailViewModel	=
70	viewModel(factory = MovieDetailViewModel.Factory)	
71) {	
72	val uiState	by
	viewModel.uiState.collectAsStateWithLifecycle()	
73	val snackbarHostState	= remember {
74	SnackbarHostState() }	
75		
76	LaunchedEffect(uiState.snackbarMessage) {	
77	uiState.snackbarMessage?.let { message ->	
78		
	snackbarHostState.showSnackbar(message, duration =	
79	SnackbarDuration.Short)	

80	viewModel.clearSnackbarMessage()
81	}
82	}
83	Scaffold(
	snackbarHost = {
	SnackbarHost(snackbarHostState) },
84	topBar = {
	TopAppBar(
85	title = { Text(uiState.movie?.title
	?: stringResource(R.string.detail_film)) },
86	navigationIcon = {
	IconButton(onClick = {
	navController.navigateUp() }) {
87	Icon(Icons.AutoMirrored.Filled.ArrowBack,
88	contentDescription = "Kembali")
	}
89	},
	colors =
90	TopAppBarDefaults.topAppBarColors(
91	containerColor =
92	MaterialTheme.colorScheme.primary,
93	titleContentColor =
94	MaterialTheme.colorScheme.onPrimary,
95	navigationIconContentColor =
96	MaterialTheme.colorScheme.onPrimary
97)
98)
99	}

100) { paddingValues ->	
	Box(
	modifier = Modifier	
	.fillMaxSize()	
101	.padding(paddingValues)	
102) {	
103	when {	
104	uiState.isLoading -> {	
105	CircularProgressIndicator(modifier	=
	Modifier.align(Alignment.Center))	
106	}	
	uiState.error != null -> {	
107	Column(
108	modifier	=
109	Modifier.align(Alignment.Center).padding(16.dp),	
	.horizontalAlignment	=
110	Alignment.CenterHorizontally,	
	.verticalArrangement	=
111	Arrangement.Center	
) {	
112	Text(
	text = "Oops!	
113	\${uiState.error}",	
114	style	=
	MaterialTheme.typography.bodyLarge,	
115	color	=
	MaterialTheme.colorScheme.error,	
116	textAlign	=
117	TextAlign.Center	

118)	
119	Spacer(modifier	=
120	Modifier.height(8.dp))	
121	Button(onClick	= {
122	viewModel.retryFetch())} {	
123	Text(stringResource(R.string.try_again))	
124	}	
125	}	
126	}	
	uiState.movie != null -> {	
127	MovieDetailContent(movie	=
	uiState.movie!!)	
128	}	
	else -> {	
129	Text(
130	text	=
131	stringResource(R.string.no_detail),	
132	modifier	=
133	Modifier.align(Alignment.Center).padding(16.dp),	
134	style	=
135	MaterialTheme.typography.bodyMedium,	
136	textAlign	=
137	TextAlign.Center	
138)	
139	}	
140	}	
141	}	
142	}	
143	}	

```

144
145 @Composable
146 fun MovieDetailContent(movie: Movie) {
147     val context = LocalContext.current
148     Column(
149         modifier = Modifier
150             .fillMaxSize()
151             .verticalScroll(rememberScrollState())
152             .padding(16.dp)
153     ) {
154         AsyncImage(
155             model =
156             ImageRequest.Builder(LocalContext.current)
157                 .data(if (movie.posterPath != null)
158                     "https://image.tmdb.org/t/p/w500${movie.posterPath}" else null)
159                 .crossfade(true)
160                 .build(),
161             placeholder = painterResource(id =
162                 R.drawable.ic_placeholder_image),
163             error = painterResource(id =
164                 R.drawable.ic_broken_image),
165             contentDescription = "Poster
166                 ${movie.title}",
167             modifier = Modifier
168                 .fillMaxWidth()
169                 .aspectRatio(2f / 3f)
170                 .clip(RoundedCornerShape(12.dp))
171             .align(Alignment.CenterHorizontally),

```

	contentScale = ContentScale.Crop	
167)	
168		
169	Spacer(modifier = Modifier.height(16.dp))	
170		
171	Text(
172	text = movie.title,	
173	style	=
174	MaterialTheme.typography.headlineMedium,	
175	fontWeight = FontWeight.Bold,	
176	textAlign = TextAlign.Center,	
	modifier = Modifier.fillMaxWidth()	
177)	
178	Spacer(modifier = Modifier.height(8.dp))	
179		
180	Row(
	modifier = Modifier.fillMaxWidth(),	
181	horizontalArrangement	=
	Arrangement.SpaceBetween,	
182	verticalAlignment	=
183	Alignment.CenterVertically	
184) {	
185	Text(
	text = "Rilis: \${movie.releaseDate	
186	?: "Tidak diketahui"}",	
	style	=
187	MaterialTheme.typography.bodyMedium,	
188	fontWeight = FontWeight.SemiBold	
189)	

190	Text(
191	text	= "Rating:
192	\${String.format("%.1f", movie.voteAverage)}/10",	
193	style	=
	MaterialTheme.typography.bodyMedium,	
194	fontWeight	= FontWeight.SemiBold
)	
195	}	
196	Spacer(modifier = Modifier.height(16.dp))	
197		
198	Text(
199	text	=
200	stringResource(R.string.overview),	
201	style	=
	MaterialTheme.typography.titleMedium,	
202	fontWeight	= FontWeight.Bold
203)	
204	Spacer(modifier = Modifier.height(4.dp))	
205	Text(
206	text	= movie.overview,
	style	=
207	MaterialTheme.typography.bodyMedium,	
208	textAlign	= TextAlign.Justify
209)	
	if (!movie.homepage.isNullOrBlank()) {	
210	Spacer(modifier	=
	Modifier.height(24.dp))	
211	Button(

212	onClick = {
	val intent =
213	Intent(Intent.ACTION_VIEW,
214	Uri.parse(movie.homepage)).apply {
	setPackage("com.android.chrome")
	}
	try {
216	context.startActivity(intent)
217	} catch (e:
218	ActivityNotFoundException) {
	val fallbackIntent =
219	Intent(Intent.ACTION_VIEW,
	Uri.parse(movie.homepage))
	try {
220	
221	context.startActivity(fallbackIntent)
222	} catch (ex:
223	ActivityNotFoundException) {
224	
225	Toast.makeText(context, "Tidak ada browser yang
226	ditemukan!", Toast.LENGTH_SHORT).show()
	}
	}
	},
227	modifier = Modifier.fillMaxWidth(),
	shape = RoundedCornerShape(12.dp)
228) {
229	

230	Icon(Icons.Default.Search,
231	contentDescription = "Homepage", modifier =
232	Modifier.size(ButtonDefaults.IconSize))
233	
234	Spacer(Modifier.size(ButtonDefaults.IconSpacing))
	Text(stringResource(R.string.visit_website))
	}
	}
	Spacer(modifier = Modifier.height(16.dp))
	}
	}

MovieItem.kt

Tabel 51. Source Code MovieItem.kt

1	package
2	com.pemrogramanmobile.apicompose.ui.screen.movieli
3	st
4	
5	import android.content.ActivityNotFoundException
6	import android.content.Intent
7	import android.net.Uri
8	import android.widget.Toast
9	import androidx.compose.foundation.layout.*
10	import
11	androidx.compose.foundation.shape.RoundedCornerSha
12	pe
13	import androidx.compose.material3.*
14	import androidx.compose.runtime.Composable

```

15 import androidx.compose.ui.Alignment
16 import androidx.compose.ui.Modifier
17 import androidx.compose.ui.draw.clip
18 import androidx.compose.ui.layout.ContentScale
19 import androidx.compose.ui.platform.LocalContext
20 import androidx.compose.ui.res.painterResource
21 import androidx.compose.ui.res.stringResource
22 import androidx.compose.ui.text.font.FontWeight
23 import androidx.compose.ui.text.style.TextAlign
24 import androidx.compose.ui.text.style.TextOverflow
25 import androidx.compose.ui.unit.dp
26 import androidx.compose.ui.unit.sp
27 import coil.compose.AsyncImage
28 import coil.request.ImageRequest
29 import com.pemrogramanmobile.apicompose.R
30 import
31 com.pemrogramanmobile.apicompose.domain.model.Movi
32 e
33
34 @Composable
35 fun MovieItem(
36     movie: Movie,
37     modifier: Modifier = Modifier,
38     onDetailsClick: (movieId: Int) -> Unit
39 ) {
40     val context = LocalContext.current
41
42     Card(
43         modifier = modifier
44         .padding(7.dp)

```


45	.fillMaxWidth()
46	.wrapContentHeight(),
47	shape = MaterialTheme.shapes.medium,
48	colors = CardDefaults.cardColors(
49	containerColor =
50	MaterialTheme.colorScheme.surface
51),
52	elevation = CardDefaults.cardElevation(
53	defaultElevation = 5.dp
54)
55) {
56	Row(
57	verticalAlignment = Alignment.Top,
58	modifier = Modifier.padding(5.dp)
59) {
60	AsyncImage(
61	model =
62	ImageRequest.Builder(LocalContext.current)
63	.data(if (movie.posterPath !=
64	null)
65	"https://image.tmdb.org/t/p/w342\${movie.posterPath
66	}" else null)
67	.crossfade(true)
68	.build(),
69	placeholder = painterResource(id =
70	R.drawable.ic_placeholder_image),
71	error = painterResource(id =
72	R.drawable.ic_broken_image),
73	contentDescription = "Poster
74	\${movie.title}",

75	modifier = Modifier	
76	.size(width = 120.dp, height =	
77	150.dp)	
78	.padding(8.dp)	
79		
80	.align(Alignment.CenterVertically)	
81		
82	.clip(RoundedCornerShape(15.dp)),	
83	contentScale	=
84	ContentScale.FillBounds,	
85)	
86	Column(
87	Modifier	
88	.weight(1f)	
89	.padding(start = 0.dp, top =	
90	8.dp, end = 8.dp, bottom = 8.dp)	
91) {	
92	Row (
93	modifier	=
94	Modifier.fillMaxWidth(),	
95	verticalAlignment	=
96	Alignment.CenterVertically	
97) {	
98	Text(
99	text = movie.title,	
100	fontSize = 20.sp,	
101	fontWeight	=
102	FontWeight.Bold,	
103	color	=
104	MaterialTheme.colorScheme.onSurface,	

105	modifier	=
106	Modifier.weight(1f),	
107	maxLines = 2,	
108	overflow	=
109	TextOverflow.Ellipsis	
110)	
111	Spacer(Modifier.width(10.dp))	
112	movie.releaseDate?.let {	
113	if (it.isNotBlank()) {	
114	Text(
115	text = it.split("-	
116	").firstOrNull() ?: it,	
117	style	=
118	MaterialTheme.typography.labelMedium,	
119	fontSize = 15.sp,	
120	color	=
121	MaterialTheme.colorScheme.onSurface	
122)	
123	}	
124	}	
125	}	
126		
127	Spacer(Modifier.height(8.dp))	
128	Text(
129	text	=
130	stringResource(R.string.overview),	
131	style	=
132	MaterialTheme.typography.labelMedium,	
133	fontWeight	=
134	FontWeight.SemiBold,	

135	color	=
136	MaterialTheme.colorScheme.onSurface	
137)	
138	Text(
139	text = movie.overview,	
140	style	=
141	MaterialTheme.typography.bodySmall,	
142	color	=
143	MaterialTheme.colorScheme.onSurface,	
144	textAlign = TextAlign.Justify,	
145	maxLines = 3,	
146	overflow	=
147	TextOverflow.Ellipsis	
148)	
149	Spacer(Modifier.height(12.dp))	
150		
151	Text(
152	text = "Rating	
153	\${String.format("%.1f", movie.voteAverage)} / 10",	
154	style	=
155	MaterialTheme.typography.bodyMedium,	
156	fontWeight	=
157	FontWeight.SemiBold,	
158	color	=
159	MaterialTheme.colorScheme.onSurface,	
160	modifier	=
161	Modifier.fillMaxWidth()	
162)	
163	Spacer(Modifier.height(12.dp))	
164		

165	Row (
166	Modifier.fillMaxWidth(),
167	horizontalArrangement =
	Arrangement.End
) {
	Button(
	onClick = {
	movie.homepage?.let {
url ->	
	if
	(url.isNotBlank()) {
	val intent =
	Intent(Intent.ACTION_VIEW, Uri.parse(url)).apply {
	setPackage("com.android.chrome")
	}
	try {
	context.startActivity(intent)
	} catch (e:
	ActivityNotFoundException) {
	val
fallbackIntent =	Intent(Intent.ACTION_VIEW,
Uri.parse(url))	
	try {
	context.startActivity(fallbackIntent)
	} catch
	(ex: ActivityNotFoundException) {

	<pre> Toast.makeText(context, "Tidak ada browser yang ditemukan!", Toast.LENGTH_SHORT).show() } } } } }, shape = RoundedCornerShape(12.dp), contentPadding = PaddingValues(horizontal = 15.dp), enabled = !movie.homepage.isNullOrBlank()) { Text(stringResource(R.string.website), fontSize = 13.sp) } Spacer(Modifier.width(10.dp)) Button(onClick = { onDetailsClick(movie.id) }, shape = RoundedCornerShape(12.dp), contentPadding = PaddingValues(horizontal = 15.dp),) { </pre>
--	---

	<pre> Text(stringResource(R.string.details), fontSize = 13.sp) } } } } } } </pre>
--	--

MovieListScreen.kt

Tabel 52. Source Code MovieListScreen.kt

1	package
2	com.pemrogramanmobile.apicompose.ui.screen.movieli
3	st
4	
5	import android.content.res.Configuration
6	import androidx.compose.foundation.layout.*
7	import androidx.compose.foundation.lazy.LazyColumn
8	import androidx.compose.foundation.lazy.items
9	import androidx.compose.material.icons.Icons
10	import
11	androidx.compose.material.icons.filled.Refresh
12	import androidx.compose.material3.*
13	import androidx.compose.runtime.*
14	import androidx.compose.ui.Alignment
15	import androidx.compose.ui.Modifier
16	import androidx.compose.ui.platform.LocalContext
17	import androidx.compose.ui.res.stringResource

18	import androidx.compose.ui.tooling.preview.Preview
19	import androidx.compose.ui.unit.dp
20	import
21	androidx.lifecycle.compose.collectAsStateWithLifecycle
22	import
23	androidx.lifecycle.viewmodel.compose.viewModel
24	import androidx.navigation.NavController
25	import
26	androidx.navigation.compose.rememberNavController
27	import
28	com.pemrogramanmobile.apicompose.MyApplication
29	import com.pemrogramanmobile.apicompose.R
30	import
31	com.pemrogramanmobile.apicompose.data.remote.RetrofitClient
32	import
33	com.pemrogramanmobile.apicompose.data.repository.MovieRepositoryImpl
34	import
35	com.pemrogramanmobile.apicompose.domain.usecase.GetPopularMoviesUseCase
36	import com.pemrogramanmobile.apicompose.ui.Routes
37	import
38	com.pemrogramanmobile.apicompose.ui.theme.ApiComposeTheme
39	import
40	com.pemrogramanmobile.apicompose.ui.viewmodel.MovieViewModel
41	
42	
43	
44	
45	
46	
47	

48	import	
49	com.pemrogramanmobile.apicompose.ui.viewmodel.Movi	
50	eViewModelFactory	
51		
52	@OptIn(ExperimentalMaterial3Api::class)	
53	@Composable	
54	fun MovieListScreen(navController: NavController) {	
55	val title = stringResource(R.string.title)	
56		
57	val	context =
58	LocalContext.current.applicationContext	
59	val movieRepositoryImpl = remember(context) {	
60	MovieRepositoryImpl(
61	tmdbApiService	=
62	RetrofitClient.instance,	
63	movieDao	= (context as
64	MyApplication).database.movieDao(),	
65	context = context	
66)	
67	}	
68	val	getPopularMoviesUseCase =
69	remember(movieRepositoryImpl) {	
70		
71	GetPopularMoviesUseCase(movieRepositoryImpl)	
72	}	
73	val movieViewModel: MovieViewModel = viewModel(
74	factory	=
75	MovieViewModelFactory(getPopularMoviesUseCase)	
76)	
77		

78	val	uiState	by
79	movieViewModel.uiState.collectAsStateWithLifecycle		
80	()		
81	val	snackbarHostState	= remember {
82	SnackbarHostState() }		
83			
84	LaunchedEffect(uiState.snackbarMessage) {		
85	uiState.snackbarMessage?.let { message ->		
86	snackbarHostState.showSnackbar(
87	message = message,		
88	duration = SnackbarDuration.Short		
89)		
90	movieViewModel.clearSnackbarMessage()		
91	}		
92	}		
93			
94	Scaffold(
95	snackbarHost	=	{
96	SnackbarHost(snackbarHostState) },		
97	topBar = {		
98	TopAppBar(
99	title = { Text(title) },		
100	colors	=	
101	TopAppBarDefaults.topAppBarColors(
102	containerColor	=	
103	MaterialTheme.colorScheme.primary,		
104	titleContentColor	=	
105	MaterialTheme.colorScheme.onPrimary		
106),		
107	actions = {		

108	IconButton(onClick	=	{
109	movieViewModel.refreshMovies() }}	{	
110	Icon(
111	Icons.Filled.Refresh,		
112	contentDescription	=	
113	"Refresh Film",		
114	tint	=	
115	MaterialTheme.colorScheme.onPrimary		
116)		
117	}		
118	}		
119)		
120	}		
121) { innerPadding ->		
122	Box(
123	modifier = Modifier		
124	.fillMaxSize()		
125	.padding(innerPadding)		
126) {		
127	if (uiState.isLoading	&&	
128	uiState.movies.isEmpty()) {		
129	CircularProgressIndicator(modifier		
130	= Modifier.align(Alignment.Center))		
131	}		
132	else if (uiState.error != null	&&	
133	uiState.movies.isEmpty()) {		
134	Column(
135	modifier	=	
136	Modifier.align(Alignment.Center).padding(16.dp),		
137			

138	horizontalAlignment	=
139	Alignment.CenterHorizontally,	
140	verticalArrangement	=
141	Arrangement.Center	
142) {	
143	Text (
144	text	= "Oops!
145	{uiState.error}",	
146	style	=
147	MaterialTheme.typography.bodyLarge,	
148	color	=
149	MaterialTheme.colorScheme.error	
150)	
151	Spacer (modifier	=
152	Modifier.height (8.dp))	
153	Button (onClick	= {
154	movieViewModel.refreshMovies())) {	
155		
156	Text (stringResource (R.string.try_again))	
157	}	
158	}	
159	}	
160	else if (uiState.movies.isNotEmpty()) {	
	LazyColumn (
	modifier	=
	Modifier.fillMaxSize(),	
	contentPadding	=
	PaddingValues (horizontal = 16.dp, vertical = 8.dp)	
) {	

	<pre> items(uiState.movies, key = { movie -> movie.id }) { movie -> MovieItem(movie = movie, modifier = Modifier.fillMaxWidth(), onDetailsClick = { movieId -> navController.navigate(Routes.navigateById(movieId)) }) Spacer(modifier = Modifier.height(8.dp)) } } else if (!uiState.isLoading && uiState.error == null) { Text(text = stringResource(R.string.no_film_found), modifier = Modifier.align(Alignment.Center).padding(16.dp), style = MaterialTheme.typography.bodyMedium) } </pre>
--	--

	<pre> if (uiState.isLoading && uiState.movies.isNotEmpty()) { LinearProgressIndicator(modifier = Modifier.fillMaxWidth().align(Alignment.TopCenter)) } } } } @Preview(showBackground = true, uiMode = Configuration.UI_MODE_NIGHT_NO) @Composable fun MovieListScreenLightPreview() { ApiComposeTheme { Surface { MovieListScreen(navController = rememberNavController()) } } } @Preview(showBackground = true, uiMode = Configuration.UI_MODE_NIGHT_YES) @Composable fun MovieListScreenDarkPreview() { ApiComposeTheme { Surface { MovieListScreen(navController = rememberNavController()) </pre>
--	--

	<pre> } } }</pre>
--	-------------------------------

MovieDetailViewModel.kt

Tabel 53. Source Code MovieDetailViewModel.kt

1	package
	com.pemrogramanmobile.apicompose.ui.viewmodel
2	
3	import androidx.lifecycle.SavedStateHandle
4	import androidx.lifecycle.ViewModel
5	import androidx.lifecycle.ViewModelProvider
6	import androidx.lifecycle.createSavedStateHandle
7	import androidx.lifecycle.viewModelScope
8	import androidx.lifecycle.viewmodel.CreationExtras
9	import
	com.pemrogramanmobile.apicompose.MyApplication
10	import
	com.pemrogramanmobile.apicompose.data.remote.RetrofitClient
11	import
	com.pemrogramanmobile.apicompose.data.repository.MovieRepositoryImpl
12	import
	com.pemrogramanmobile.apicompose.domain.model.Movie
13	e
	import
	com.pemrogramanmobile.apicompose.domain.usecase.GetMovieDetailsUseCase
14	

	import
15	com.pemrogramanmobile.apicompose.util.NetworkResul
16	t
17	import kotlinx.coroutines.flow.MutableStateFlow
18	import kotlinx.coroutines.flow.StateFlow
19	import kotlinx.coroutines.flow.asStateFlow
20	import kotlinx.coroutines.flow.launchIn
21	import kotlinx.coroutines.flow.onEach
22	import kotlinx.coroutines.flow.update
23	
24	data class MovieDetailUiState(
25	val movie: Movie? = null,
26	val isLoading: Boolean = false,
27	val error: String? = null,
28	val snackbarMessage: String? = null
29)
30	
	class MovieDetailViewModel(
31	private val getMovieDetailsUseCase:
32	GetMovieDetailsUseCase,
33	private val savedStateHandle: SavedStateHandle
34) : ViewModel() {
	private val _uiState =
35	MutableStateFlow(MovieDetailUiState(isLoading =
	true))
36	val uiState: StateFlow<MovieDetailUiState> =
37	_uiState.asStateFlow()

38	private val movieId: Int =
39	savedStateHandle.get<String>("movieId")?.toIntOrNull()
40	?: 0
41	
42	init {
43	if (movieId != 0) {
44	fetchMovieDetails(movieId)
45	} else {
46	_uiState.update {
47	it.copy(isLoading = false, error =
48	"Movie ID tidak valid.")
49	}
50	}
51	private fun fetchMovieDetails(id: Int) {
52	_uiState.update { it.copy(isLoading = true,
53	error = null, snackbarMessage = null) }
54	
55	getMovieDetailsUseCase(id)
56	.onEach { result ->
57	_uiState.update { currentState ->
58	when (result) {
59	is NetworkResult.Success ->
60	{
61	currentState.copy(
62	movie =
63	result.data,
64	isLoading = false,
	error = null

65)
	}
66	is NetworkResult.Error -> {
67	val errorMessage =
	result.message ?: "Terjadi kesalahan tidak
68	diketahui"
	if (currentState.movie
69	!= null) {
70	currentState.copy(
71	isLoading =
72	false,
73	snackbarMessage = errorMessage
74)
	} else {
75	currentState.copy(
76	movie = null,
77	isLoading =
78	false,
79	error =
80	errorMessage
81)
82	}
83	}
84	}
85	}
	}
86	.launchIn(viewModelScope)
87	}
88	

89	fun clearSnackbarMessage() {
90	_uiState.update { it.copy(snackbarMessage =
91	null) }
92	}
93	
94	fun retryFetch() {
95	if (movieId != 0) {
	fetchMovieDetails(movieId)
96	}
97	}
98	
99	companion object {
100	val Factory: ViewModelProvider.Factory =
101	object : ViewModelProvider.Factory {
	@Suppress("UNCHECKED_CAST")
	override fun <T : ViewModel> create(
102	modelClass: Class<T>,
	extras: CreationExtras
103): T {
104	val application =
	checkNotNull(extras[ViewModelProvider.AndroidViewM
105	odelFactory.APPLICATION_KEY]) as MyApplication
106	val savedStateHandle =
107	extras.createSavedStateHandle()
108	
109	val movieRepository =
	MovieRepositoryImpl(
110	RetrofitClient.instance,
111	
112	application.database.movieDao(),

113	application.applicationContext
114)
115	val getMovieDetailsUseCase =
116	GetMovieDetailsUseCase(movieRepository)
117	
118	return MovieDetailViewModel(
	getMovieDetailsUseCase,
	savedStateHandle
) as T
	}
	}
	}
	}

MovieViewModel.kt

Tabel 54. Source Code MovieViewModel.kt

1	package
	com.pemrogramanmobile.apicompose.ui.viewmodel
2	
3	import androidx.lifecycle.ViewModel
4	import androidx.lifecycle.ViewModelProvider
5	import androidx.lifecycle.viewModelScope
6	import
	com.pemrogramanmobile.apicompose.domain.model.Movi
7	e
	import
	com.pemrogramanmobile.apicompose.domain.usecase.Ge
8	tPopularMoviesUseCase

9	import
10	com.pemrogramanmobile.apicompose.util.NetworkResult
11	
12	import kotlinx.coroutines.flow.MutableStateFlow
13	import kotlinx.coroutines.flow.StateFlow
14	import kotlinx.coroutines.flow.asStateFlow
15	import kotlinx.coroutines.flow.launchIn
16	import kotlinx.coroutines.flow.onEach
17	import kotlinx.coroutines.flow.update
18	
19	data class MovieListUiState(
20	val movies: List<Movie> = emptyList(),
21	val isLoading: Boolean = false,
22	val error: String? = null,
23	val snackbarMessage: String? = null
24)
25	class MovieViewModel(
26	private val getPopularMoviesUseCase: GetPopularMoviesUseCase
27) : ViewModel() {
28	private val _uiState =
29	MutableStateFlow(MovieListUiState(isLoading =
30	true))
31	val uiState: StateFlow<MovieListUiState> =
32	_uiState.asStateFlow()
33	init {
34	fetchPopularMovies()

	}
35	
	private fun fetchPopularMovies(forceRefresh:
36	Boolean = false) {
	if (forceRefresh
37	_uiState.value.movies.isEmpty()) {
38	_uiState.update { it.copy(isLoading =
39	true, error = null, snackbarMessage = null) }
40	}
41	
42	getPopularMoviesUseCase()
43	.onEach { result ->
44	_uiState.update { currentState ->
45	when (result) {
	is NetworkResult.Success ->
46	{
47	currentState.copy(
48	movies =
	result.data ?: emptyList(),
49	isLoading = false,
50	error = null,
51	snackbarMessage =
52	null
)
53	}
	is NetworkResult.Error -> {
54	val errorMessage =
55	result.message ?: "Terjadi kesalahan tidak diketahui"
56	

	if
57	(result.data.isNullOrEmpty()) {
	currentState.copy(
58	movies =
	emptyList(),
59	isLoading =
60	false,
61	error =
62	errorMessage,
63	snackbarMessage = null
)
64	} else {
65	currentState.copy(
	movies =
66	result.data,
67	isLoading =
68	false,
69	error = null,
70	
71	snackbarMessage = errorMessage
72)
73	}
74	}
75	}
76	}
	}
77	.launchIn(viewModelScope)
78	}
79	

80	fun clearSnackbarMessage() {
81	_uiState.update { it.copy(snackbarMessage =
82	null) }
83	}
84	
85	fun refreshMovies() {
	fetchPopularMovies(forceRefresh = true)
86	}
87	}
88	class MovieViewModelFactory(
	private val getPopularMoviesUseCase:
	GetPopularMoviesUseCase
89) : ViewModelProvider.Factory {
90	override fun <T : ViewModel> create(modelClass:
	Class<T>): T {
91	if
92	(modelClass.isAssignableFrom(MovieViewModel::class
	.java)) {
93	@Suppress("UNCHECKED_CAST")
94	return
	MovieViewModel(getPopularMoviesUseCase) as T
	}
	throw IllegalArgumentException("Unknown
	ViewModel class")
	}
	}

AppNavigation.kt

Tabel 55. Source Code AppNavigation.kt

1	package com.pemrogramanmobile.apicompose.ui
2	
3	import androidx.compose.runtime.Composable
4	import androidx.navigation.NavType
5	import androidx.navigation.compose.NavHost
6	import androidx.navigation.compose.composable
7	import
	androidx.navigation.compose.rememberNavController
8	import androidx.navigation.navArgument
9	import
	com.pemrogramanmobile.apicompose.ui.screen.moviede
	tail.MovieDetailScreen
10	import
	com.pemrogramanmobile.apicompose.ui.screen.movieli
	st.MovieListScreen
11	
12	@Composable
13	fun AppNavigation() {
14	val navController = rememberNavController()
15	NavHost(navController = navController,
	startDestination = Routes.MOVIE_LIST) {
16	composable(Routes.MOVIE_LIST) {
17	MovieListScreen(navController =
	navController)
18	}
19	composable(
20	route = Routes.MOVIE_DETAIL,
21	

	arguments	=
22	listOf(navArgument("movieId") { type	=
23	NavType.StringType })	
) {	
24	MovieDetailScreen(navController	=
25	navController)	
26	}	
	}	
	}	

Routes.kt

Tabel 56. Source Code Routes.kt

1	package com.pemrogramanmobile.apicompose.ui
2	
3	object Routes {
4	const val MOVIE_LIST = "movieList"
5	const val MOVIE_DETAIL = "movieDetail/{movieId}"
6	
7	fun navigateById(id: Int): String{
8	return "movieDetail/\${id}"
9	}
10	}

AppSettings.kt

Tabel 57. Source Code AppSettings.kt

1	package com.pemrogramanmobile.apicompose.util
2	
3	import android.content.Context

4	import android.content.SharedPreferences
5	
6	object AppSettings {
7	private const val PREFS_NAME =
	"app_settings_prefs"
8	private const val
	KEY_LAST_CACHE_TIMESTAMP_MOVIES =
9	"last_cache_timestamp_movies"
10	private const val KEY_UI_THEME = "ui_theme"
11	
	private fun getPrefs(context: Context):
12	SharedPreferences {
	return
	context.getSharedPreferences(PREFS_NAME,
13	Context.MODE_PRIVATE)
14	}
15	
	fun getLastMoviesCacheTimestamp(context:
16	Context): Long {
	return
	getPrefs(context).getLong(KEY_LAST_CACHE_TIMESTAMP
17	_MOVIES, 0L)
18	}
19	
	fun setLastMoviesCacheTimestamp(context:
	Context, timestamp: Long) {
20	
	getPrefs(context).edit().putLong(KEY_LAST_CACHE_TI
21	MESTAMP_MOVIES, timestamp).apply()
22	}

23	fun getTheme(context: Context): String {
24	return
	getPrefs(context).getString(KEY_UI_THEME, "system")
	?: "system"
25	}
26	
27	fun setTheme(context: Context, theme: String) {
28	getPrefs(context).edit().putString(KEY_UI_THEME,
	theme).apply()
29	}
30	}

NetworkResult.kt

Tabel 58. Source Code NetworkResult.kt

1	package com.pemrogramanmobile.apicompose.util
2	
3	sealed class NetworkResult<T>{
4	val data: T? = null,
5	val message: String? = null
6) {
7	class Success<T>(data: T) :
	NetworkResult<T>(data)
8	class Error<T>(message: String, data: T? = null)
	: NetworkResult<T>(data, message)
9	}

MainActivity.kt

Tabel 59. Source Code MainActivity.kt

1	package com.pemrogramanmobile.apicompose
2	
3	import android.os.Bundle
4	import androidx.activity.ComponentActivity
5	import androidx.activity.compose.setContent
6	import
	androidx.compose.foundation.layout.fillMaxSize
7	import androidx.compose.material3.MaterialTheme
8	import androidx.compose.material3.Surface
9	import androidx.compose.ui.Modifier
10	import
	com.pemrogramanmobile.apicompose.ui.AppNavigation
11	import
	com.pemrogramanmobile.apicompose.ui.theme.ApiComposeTheme
12	
13	class MainActivity : ComponentActivity() {
14	override fun onCreate(savedInstanceState: Bundle?) {
15	super.onCreate(savedInstanceState)
16	setContent {
17	ApiComposeTheme {
18	Surface(
19	modifier =
	Modifier.fillMaxSize(),
20	color =
	MaterialTheme.colorScheme.background
21) {
22	AppNavigation()

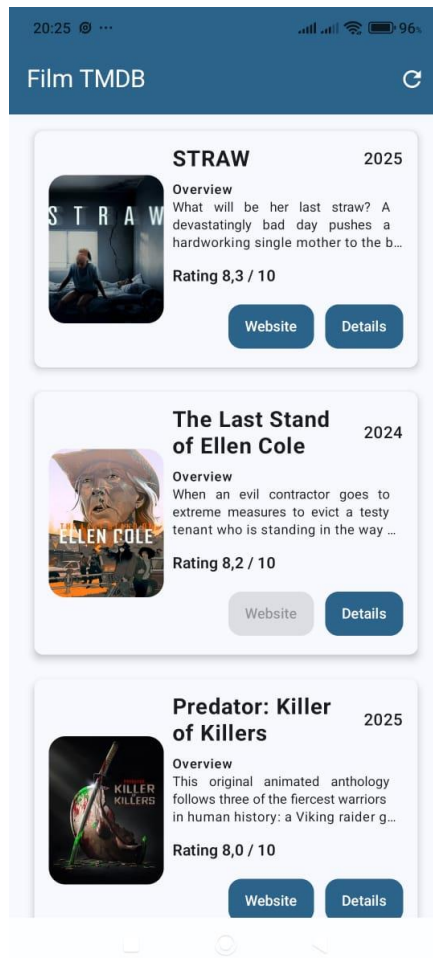
23	}
24	}
25	}
26	}
27	}

MyApplication.kt

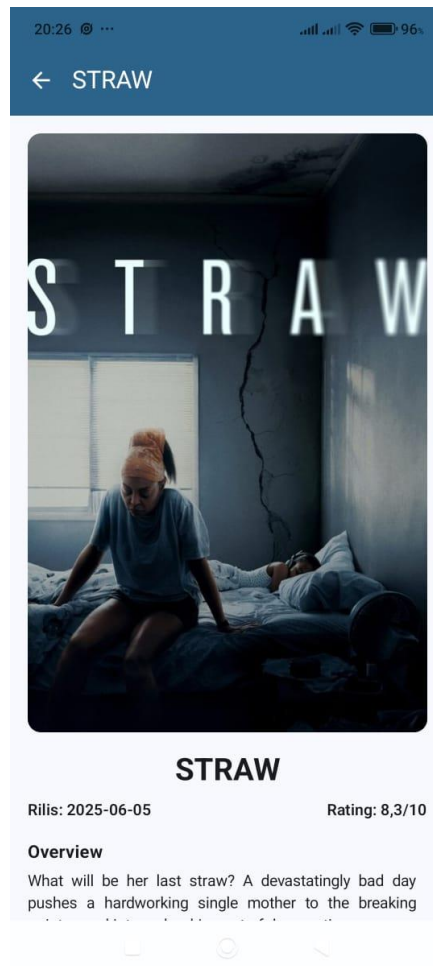
Tabel 60. Source Code MyApplication.kt

1	package com.pemrogramanmobile.apicompose
2	
3	import android.app.Application
4	import com.pemrogramanmobile.apicompose.data.local.AppDat abase
5	
6	class MyApplication : Application() {
7	val database: AppDatabase by lazy { AppDatabase.getDatabase(this) }
8	}

B. Output Program



Gambar 30. Screenshot List Screen



Gambar 31. Screenshot Detail Screen

C. Pembahasan

`/data/local/AppDatabase.kt`

File ini mendefinisikan kelas `AppDatabase` yang merupakan inti dari implementasi Room. Kelas ini diannotasi dengan `@Database` yang memberitahu Room bahwa ini adalah konfigurasi database. `entities = [MovieEntity::class]` mendaftarkan tabel `MovieEntity` ke dalam database. `version = 1` menandakan versi skema database, yang penting untuk migrasi (meskipun di sini `fallbackToDestructiveMigration()` digunakan, yang akan membuat ulang database jika versi berubah, menghapus data

lama; ini praktis untuk pengembangan tetapi memerlukan strategi migrasi yang lebih hati-hati untuk produksi). `exportSchema = false` biasanya digunakan untuk menonaktifkan ekspor skema JSON yang bisa berguna untuk histori versi. `AppDatabase` adalah kelas abstrak yang mewarisi `RoomDatabase` dan menyediakan akses ke DAO (Data Access Object) melalui fungsi abstrak `movieDao()`. Companion object-nya mengimplementasikan pola Singleton untuk memastikan hanya ada satu instance database (INSTANCE) di seluruh aplikasi, yang dibuat menggunakan `Room.databaseBuilder()`. Ini penting untuk efisiensi dan konsistensi data.

`/data/local/MovieDao.kt`

`MovieDao` (Data Access Object) adalah interface yang diannotasi dengan `@Dao`. `Room` akan mengimplementasikan metode-metode di dalamnya secara otomatis. File ini mendefinisikan operasi database untuk entitas `MovieEntity`.

- `@Insert(onConflict = OnConflictStrategy.REPLACE)`: Menyisipkan daftar film. Jika ada film dengan `PrimaryKey` yang sama, film lama akan diganti. Ini berguna untuk menjaga data tetap up-to-date.
- `@Query("SELECT * FROM movies ORDER BY voteAverage DESC")` fun `getAllMovies(): Flow<List<MovieEntity>>`: Mengambil semua film dari tabel `movies`, diurutkan berdasarkan `voteAverage` secara menurun. Penggunaan `Flow<List<MovieEntity>>` sangat penting karena memungkinkan UI untuk secara reaktif mengamati perubahan data di database. Setiap kali data di tabel `movies` berubah, `Flow` ini akan memancarkan daftar film yang terbaru.
- `@Query("SELECT * FROM movies WHERE id = :movieId")` fun `getMovieById(movieId: Int): Flow<MovieEntity?>`: Mengambil satu film berdasarkan ID-nya, juga menggunakan `Flow` untuk pembaruan reaktif.

- `@Query("DELETE FROM movies") suspend fun deleteAllMovies():` Menghapus semua data film dari tabel. Ini digunakan dalam caching strategy untuk membersihkan cache lama.
- `@Query("SELECT COUNT(*) FROM movies") suspend fun getMovieCount(): Int:` Mengambil jumlah film yang tersimpan. Ini juga bagian dari logika caching strategy.
- `@Query("SELECT lastRefreshed FROM movies ORDER BY lastRefreshed DESC LIMIT 1") suspend fun getLastRefreshTimestamp(): Long?:` Mengambil timestamp kapan terakhir kali data film di-refresh. Ini adalah komponen kunci dari caching strategy yang diimplementasikan, untuk menentukan apakah cache masih valid atau sudah kedaluwarsa.

/data/local/MovieEntity.kt

File ini mendefinisikan `MovieEntity`, sebuah data class yang diannotasi dengan `@Entity(tableName = "movies")`. Ini merepresentasikan tabel `movies` dalam database Room.

- `@PrimaryKey val id: Int:` Menandakan bahwa `id` adalah kunci utama tabel.
- Kolom-kolom lain seperti `title`, `overview`, `posterPath`, `releaseDate`, `voteAverage`, dan `homepage` menyimpan detail film.
- `val lastRefreshed: Long = System.currentTimeMillis():` Kolom ini sangat penting untuk caching strategy. Setiap kali entitas film disimpan atau diperbarui, kolom ini idealnya akan diisi dengan timestamp saat itu, yang kemudian digunakan oleh `MovieDao` dan `MovieRepositoryImpl` untuk menentukan usia cache.

/data/remote/MovieApiModel.kt

File ini berisi dua data class, `MovieListResponse` dan `MovieApiModel`, yang keduanya diannotasi dengan `@Serializable`. Ini menunjukkan penggunaan `KotlinX Serialization` sebagai library JSON untuk mengonversi respons JSON dari TMDB API menjadi objek Kotlin, dan sebaliknya jika diperlukan.

`MovieListResponse`: Merepresentasikan struktur respons JSON ketika meminta daftar film populer dari API. Ini mencakup informasi paginasi (`page`, `totalPages`, `totalResults`) dan daftar film (`results` yang merupakan `List<MovieApiModel>`).

`MovieApiModel`: Merepresentasikan satu objek film seperti yang diterima dari TMDB API. `@SerializedName("poster_path")` digunakan untuk memetakan nama field JSON (misalnya `poster_path`) ke nama properti Kotlin yang berbeda (misalnya `posterPath`) jika ada perbedaan penamaan. Properti `homepage` juga disertakan, yang penting untuk detail film.

`/data/remote/RetrofitClient.kt`

`RetrofitClient` adalah object (Singleton) yang bertanggung jawab untuk mengkonfigurasi dan menyediakan instance `Retrofit`. Ini adalah inti dari implementasi networking library `Retrofit`.

- `BASE_URL`: URL dasar untuk TMDB API.
- `API_KEY`: Kunci API Anda untuk TMDB. Catatan: Sebaiknya kunci API tidak di-hardcode seperti ini dalam kode produksi; pertimbangkan untuk menyimpannya di `gradle.properties` atau menggunakan teknik lain yang lebih aman.
- `Json { ignoreUnknownKeys = true; coerceInputValues = true }`: Konfigurasi untuk `KotlinX Serialization`. `ignoreUnknownKeys = true` membuat parser lebih toleran terhadap field JSON tambahan yang mungkin tidak didefinisikan di `MovieApiModel`. `coerceInputValues = true` membantu menangani nilai default

jika ada tipe data yang tidak cocok atau null dari API, namun harus digunakan dengan hati-hati.

- `HttpLoggingInterceptor`: Digunakan untuk mencatat request dan response HTTP, sangat berguna untuk debugging masalah jaringan. Levelnya diatur ke `BODY` untuk melihat detail penuh.
- `OkHttpClient.Builder()`: Mengkonfigurasi klien HTTP. Sebuah interceptor ditambahkan untuk secara otomatis menyisipkan `api_key` sebagai query parameter ke setiap request yang dikirim ke TMDB API.
- `Retrofit.Builder()`: Membangun instance Retrofit dengan `baseUrl`, `httpClient` yang sudah dikonfigurasi, dan yang penting, `addConverterFactory(json.asConverterFactory("application/json".toMediaType()))`. Baris ini mengintegrasikan `KotlinX Serialization` dengan Retrofit sehingga Retrofit dapat secara otomatis mengonversi respons JSON menjadi objek Kotlin (`MovieApiModel`, `MovieListResponse`) dan sebaliknya.
- `instance: TmdbApiService by lazy { ... }`: Menyediakan instance `TmdbApiService` secara lazy, artinya instance Retrofit dan service baru akan dibuat saat pertama kali diakses.

`/data/remote/TmdbApiService.kt`

Ini adalah interface yang mendefinisikan endpoints TMDB API yang akan diakses oleh aplikasi menggunakan Retrofit.

- `@GET("movie/popular") suspend fun getPopularMovies(...)`: Mendefinisikan request GET ke endpoint `movie/popular` untuk mendapatkan daftar film populer. `@Query("page")` menambahkan parameter halaman ke request. Fungsi ini adalah suspend fun karena operasi jaringan harus dijalankan secara asinkron menggunakan Kotlin Coroutines. Ia mengembalikan `MovieListResponse`.
- `@GET("movie/{movie_id}") suspend fun getMovieDetails(...)`: Mendefinisikan request GET ke endpoint `movie/{movie_id}` untuk

mendapatkan detail film tertentu. `@Path("movie_id")` menggantikan `{movie_id}` dalam URL dengan nilai parameter `movieId`. Fungsi ini juga suspend fun dan mengembalikan `MovieApiModel`.

/data/repository/MovieRepositoryImpl.kt

File ini adalah implementasi konkret dari `MovieRepository`. Ini adalah komponen kunci yang bertindak sebagai Single Source of Truth untuk data film, mengelola pengambilan data dari API (`tmdbApiService`) dan cache lokal (`movieDao`).

- Konstruktor menerima `TmdbApiService`, `MovieDao`, dan `Context`.
- `CACHE_EXPIRY_MS = TimeUnit.HOURS.toMillis(1)`: Mendefinisikan durasi cache selama 1 jam. Ini adalah bagian dari caching strategy.
- `getPopularMovies()`: `Flow<NetworkResult<List<Movie>>>`:
 - Menggunakan Flow untuk memancarkan data secara asinkron.
 - Caching Strategy:
 - 1) Memeriksa timestamp refresh terakhir (`lastRefreshTime`) dari DAO dan jumlah film di cache (`movieCount`).
 - 2) Menghitung apakah cache sudah basi (`isCacheStale`) berdasarkan `CACHE_EXPIRY_MS`.
 - 3) Jika cache tidak kosong dan belum basi, data dari `movieDao.getAllMovies()` diambil, di-map ke `List<Movie>`, dan dipancarkan sebagai `NetworkResult.Success`.
 - 4) Jika cache kosong atau sudah basi, maka:
 - 5) Dilakukan pemanggilan API ke `tmdbApiService.getPopularMovies()`.
 - 6) Data dari API (hasil `movieListResponse.results`) diubah menjadi `List<MovieEntity>` menggunakan `toMovieEntities()`, yang juga menyematkan `currentTime` sebagai `lastRefreshed`.
 - 7) Cache lama dihapus dengan `movieDao.deleteAllMovies()`.

- 8) Data baru dari API disimpan ke Room dengan `movieDao.insertMovies()`.
 - 9) Data segar kemudian diambil dari `movieDao.getAllMovies()` dan dipancarkan sebagai `NetworkResult.Success`.
 - 10) Error Handling: Jika terjadi exception saat mengambil data dari jaringan, ia akan mencoba memancarkan data yang mungkin masih ada di cache sebagai fallback (`NetworkResult.Error` dengan data fallback). Jika bahkan pengambilan dari database setelah refresh gagal, `NetworkResult.Error` juga dipancarkan.
- Penggunaan `NetworkResult`: Membungkus hasil operasi (sukses dengan data, atau error dengan pesan dan mungkin data lama/parsial) memberikan cara yang terstruktur untuk menangani status panggilan API dan operasi data di lapisan atas (`ViewModel`).
 - `getMovieDetails(movieId: Int): Flow<NetworkResult<Movie>>`:
 - Pertama, mencoba mengambil data dari cache (`movieDao.getMovieById(movieId)`). Jika ada, data tersebut langsung dipancarkan sebagai `NetworkResult.Success`.
 - Kemudian, selalu mencoba mengambil detail film terbaru dari API (`tmdbApiService.getMovieDetails(movieId)`).
 - Data dari API diubah menjadi `MovieEntity` dan disimpan/diperbarui di Room menggunakan `movieDao.insertMovies(listOf(movieEntity))`. Pembaruan ini penting karena `MovieEntity` memiliki `OnConflictStrategy.REPLACE`, jadi data film akan diperbarui jika sudah ada.
 - Data terbaru dari API (yang sudah di-map ke `Movie`) dipancarkan sebagai `NetworkResult.Success`.
 - Error Handling: Jika API gagal dan tidak ada data di cache, `NetworkResult.Error` tanpa data dipancarkan. Jika API gagal tetapi ada data di cache, `NetworkResult.Error` dipancarkan dengan pesan error dan data cache sebelumnya.

- Caching Strategy untuk Detail Film: Strategi di sini adalah cache-first, then network and update cache. Selalu mencoba menyajikan data dari cache terlebih dahulu untuk respons cepat, lalu mengambil data terbaru dari jaringan untuk menyegarkan cache dan UI. Properti homepage yang mungkin tidak ada di daftar populer akan diisi dari sini.

Penjelasan Caching Strategy yang digunakan:

Strategi caching yang digunakan adalah kombinasi dari:

- Time-based Expiry untuk Daftar Film Populer: Data daftar film populer dianggap stale (basi) setelah 1 jam (CACHE_EXPIRY_MS). Jika cache belum basi dan tidak kosong, data dari cache akan disajikan. Jika sudah basi atau cache kosong, data baru akan diambil dari jaringan, cache akan dihapus (deleteAllMovies) dan diisi ulang dengan data baru. Penghapusan semua film sebelum memasukkan yang baru adalah pendekatan sederhana untuk daftar "populer" yang mungkin berubah secara keseluruhan.
- Cache-First, then Network and Update untuk Detail Film: Untuk detail film, aplikasi pertama-tama mencoba memuat dari cache. Kemudian, ia selalu berusaha mengambil versi terbaru dari jaringan. Data yang berhasil diambil dari jaringan akan memperbarui entri di cache. Ini memastikan pengguna melihat data terbaru jika tersedia, sambil tetap memiliki data cache jika jaringan gagal setelah pemuatan awal.

Alasan penggunaan strategi ini:

- Keseimbangan Kinerja dan Keterkinian: Untuk daftar film populer, data tidak harus selalu real-time detik demi detik. Periode kadaluwarsa 1 jam memberikan keseimbangan yang baik antara mengurangi panggilan API berlebih dan menjaga data tetap cukup segar.
- Pengalaman Pengguna Offline/Lambat: Menyajikan data dari cache (jika ada) meningkatkan pengalaman pengguna saat koneksi internet lambat atau tidak tersedia.

- Efisiensi Sumber Daya: Mengurangi jumlah panggilan jaringan menghemat baterai dan penggunaan data pengguna.
- Pembaruan Detail Spesifik: Untuk detail film, penting untuk mendapatkan informasi selengkap mungkin (termasuk homepage), sehingga pemanggilan jaringan setelah pemuatan cache dibenarkan untuk menyegarkan dan melengkapi data.

/data/Mappers.kt

File ini berisi serangkaian fungsi ekstensi untuk memetakan (mengubah) objek dari satu layer/model ke layer/model lain. Ini adalah praktik yang baik untuk menjaga pemisahan antar layer.

- `MovieApiModel.toDomainMovie()`: Mengubah objek API menjadi objek Domain (Movie).
- `MovieApiModel.toMovieEntity(lastRefreshedTime: Long)`: Mengubah objek API menjadi objek Entity Room (MovieEntity), dengan menyertakan `lastRefreshedTime` yang penting untuk caching.
- `MovieEntity.toDomainMovie()`: Mengubah objek Entity Room menjadi objek Domain.
- Fungsi-fungsi untuk mengubah List dari satu tipe ke tipe lain (misalnya, `List<MovieApiModel>.toDomainMovies()`).

/domain/model/Movie.kt

Ini adalah data class `Movie` yang merepresentasikan entitas film di domain layer. Model ini yang akan digunakan oleh UI dan Use Case. Model ini lebih sederhana dan hanya berisi data yang relevan untuk logika bisnis dan presentasi, terlepas dari bagaimana data itu disimpan atau darimana asalnya. Field `homepage` juga ada di sini.

/domain/repository/MovieRepository.kt

Ini adalah interface yang mendefinisikan kontrak untuk `MovieRepositoryImpl`. Ini adalah bagian dari prinsip `Dependency Inversion`, di mana domain layer mendefinisikan bagaimana ia ingin berinteraksi dengan data, tanpa mengetahui detail implementasi (apakah data berasal dari API, database, atau keduanya).

- Metode-metodenya (`getPopularMovies`, `getMovieDetails`) mengembalikan `Flow<NetworkResult<...>>`, yang menunjukkan bahwa mereka menyediakan aliran data asinkron yang juga mencakup informasi status (sukses/error).

`/domain/usecase/GetMovieDetailsUseCase.kt`

dan `/domain/usecase/GetPopularMoviesUseCase.kt`

Kedua file ini mendefinisikan use case (atau interactor). Use case merangkum satu unit logika bisnis atau fungsionalitas aplikasi.

- Masing-masing mengambil `MovieRepository` sebagai dependensi.
- Menggunakan operator `fun invoke` memungkinkan instance use case dipanggil seolah-olah itu adalah fungsi. Contoh: `getPopularMoviesUseCase()`.
- Mereka hanya mendelegasikan panggilan ke metode yang sesuai di `movieRepository`. Dalam aplikasi yang lebih kompleks, use case mungkin berisi logika tambahan.

`/ui/screen/moviedetail/MovieDetailScreen.kt`

Ini adalah `Composable` yang bertanggung jawab untuk menampilkan detail satu film.

- Menggunakan `viewModel(factory = MovieDetailViewModel.Factory)` untuk mendapatkan instance `MovieDetailViewModel`.
- `uiState` by `viewModel.uiState.collectAsStateWithLifecycle()`: Mengamati `StateFlow` dari `ViewModel` untuk memperbarui UI secara reaktif.
- `SnackbarHostState` dan `LaunchedEffect` digunakan untuk menampilkan pesan `Snackbar` dari `ViewModel`.

- Scaffold dengan TopAppBar yang menampilkan judul film dan tombol kembali.
- Logika kondisional (when) untuk menampilkan CircularProgressIndicator saat isLoading, pesan error jika uiState.error != null, konten detail jika uiState.movie != null, atau pesan default jika tidak ada data.
- MovieDetailContent: Composable terpisah untuk menampilkan konten detail:
 - AsyncImage (dari library Coil) digunakan untuk memuat dan menampilkan gambar poster film dari URL ([https://image.tmdb.org/t/p/w500\\${movie.posterPath}](https://image.tmdb.org/t/p/w500${movie.posterPath})). Ini menangani pemuatan gambar secara asinkron, caching gambar, dan menampilkan placeholder atau gambar error.
 - Menampilkan judul, tanggal rilis, rating, overview.
 - Tombol "Kunjungi Website" akan muncul jika movie.homepage ada dan tidak kosong. Tombol ini menggunakan Intent(Intent.ACTION_VIEW, Uri.parse(movie.homepage)) untuk membuka URL di browser. Ada logika fallback jika Chrome tidak terinstal.

/ui/screen/movielist/MovieItem.kt

Composable ini mendefinisikan tampilan untuk satu item film dalam daftar.

- Menggunakan Card untuk tampilan item.
- AsyncImage (dari Coil) digunakan untuk memuat poster film, dengan ukuran yang lebih kecil (w342) dibandingkan layar detail.
- Menampilkan judul, tahun rilis (diekstrak dari releaseDate), overview singkat (dengan maxLines = 3), dan rating.
- Terdapat dua tombol:
 - "Website": Mirip dengan di MovieDetailScreen, membuka movie.homepage jika tersedia.
 - "Details": Memanggil onDetailsClick(movie.id) yang akan menavigasi ke layar detail film.

/ui/screen/movielist/MovieListScreen.kt

Composable utama untuk menampilkan daftar film populer.

- Menginisialisasi `MovieRepositoryImpl` dan `GetPopularMoviesUseCase`. Catatan: Idealnya, dependensi seperti ini akan di-provide menggunakan library `Dependency Injection` seperti `Hilt` atau `Koin`, daripada dibuat secara manual di `Composable`. Namun, untuk proyek yang lebih kecil, pendekatan ini masih bisa diterima.
- Menggunakan `viewModel(factory = MovieViewModelFactory(getPopularMoviesUseCase))` untuk mendapatkan `MovieViewModel`.
- Mengamati `uiState` dari `MovieViewModel`.
- Scaffold dengan `TopAppBar` yang menampilkan judul dan ikon refresh (`IconButton` yang memanggil `movieViewModel.refreshMovies()`).
- Logika kondisional untuk menampilkan:
 - `CircularProgressIndicator` jika `isLoading` dan daftar film kosong.
 - Pesan error dan tombol "Coba Lagi" jika ada error dan daftar kosong.
 - `LazyColumn` untuk menampilkan daftar `uiState.movies` menggunakan `MovieItem` jika daftar tidak kosong. `key = { movie -> movie.id }` penting untuk performa `LazyColumn`.
 - Pesan "Tidak ada film yang ditemukan" jika tidak loading, tidak ada error, dan daftar kosong.
 - `LinearProgressIndicator` di bagian atas jika `isLoading` tetapi daftar film sudah ada (menandakan refresh di latar belakang).
- `onDetailsClick` pada `MovieItem` menavigasi ke rute `"movieDetail/$movieId"` menggunakan `navController`.
- Terdapat `Composable Preview` (`MovieListScreenLightPreview` dan `MovieListScreenDarkPreview`) untuk memudahkan pengembangan UI.

/ui/viewmodel/MovieDetailViewModel.kt

ViewModel untuk `MovieDetailScreen`.

- `MovieDetailUiState`: Data class yang merepresentasikan state UI untuk layar detail (data film, status loading, error, pesan snackbar).
- Konstruktor menerima `GetMovieDetailsUseCase` dan `SavedStateHandle`. `SavedStateHandle` digunakan untuk mengambil `movieId` yang diteruskan melalui argumen navigasi.
- `_uiState` adalah `MutableStateFlow` (privat) dan `uiState` adalah `StateFlow` (publik, read-only) yang diekspos ke UI.
- `init` block: Memeriksa apakah `movieId` valid. Jika ya, panggil `fetchMovieDetails()`. Jika tidak, set state error.
- `fetchMovieDetails(id: Int)`:
 - Memperbarui `_uiState` untuk menandakan `isLoading = true`.
 - Memanggil `getMovieDetailsUseCase(id)` yang mengembalikan `Flow<NetworkResult<Movie>>`.
 - Menggunakan `.onEach { result -> ... }` untuk memproses setiap emisi dari `Flow`:
 - Jika `NetworkResult.Success`, perbarui state dengan data film dan set `isLoading = false`.
 - Jika `NetworkResult.Error`, perbarui state. Jika sudah ada data film sebelumnya (misalnya dari cache), tampilkan pesan error sebagai `snackbarMessage`. Jika belum ada data film, tampilkan sebagai error utama di layar.
 - `.launchIn(viewModelScope)` meluncurkan koleksi `Flow` dalam `viewModelScope`, yang akan otomatis dibatalkan saat `ViewModel` dihancurkan.
- `clearSnackbarMessage()`: Untuk membersihkan pesan snackbar setelah ditampilkan.
- `retryFetch()`: Untuk mencoba lagi mengambil data jika terjadi error.
- Companion object `Factory`: Implementasi `ViewModelProvider.Factory` untuk membuat instance `MovieDetailViewModel` dengan dependensi yang diperlukan (`GetMovieDetailsUseCase` yang dibuat dengan `MovieRepositoryImpl`). `application.database.movieDao()` diambil dari `MyApplication`.

/ui/viewmodel/MovieViewModel.kt

ViewModel untuk MovieListScreen.

- `MovieListUiState`: Data class untuk state UI daftar film.
- Konstruktor menerima `GetPopularMoviesUseCase`.
- Sama seperti `MovieDetailViewModel`, menggunakan `MutableStateFlow` dan `StateFlow` untuk `uiState`.
- `init` block: Memanggil `fetchPopularMovies()` saat ViewModel dibuat.
- `fetchPopularMovies(forceRefresh: Boolean = false)`:
 - Jika `forceRefresh` (misalnya dari tombol refresh) atau daftar film masih kosong, set `isLoading = true`.
 - Memanggil `getPopularMoviesUseCase()` dan memproses `NetworkResult`:
 - `NetworkResult.Success`: Perbarui state dengan daftar film.
 - `NetworkResult.Error`: Sama seperti di `MovieDetailViewModel`, tampilkan error sebagai `snackbarMessage` jika sudah ada data, atau sebagai error utama jika belum ada.
- `clearSnackBarMessage()` dan `refreshMovies()` (yang memanggil `fetchPopularMovies(forceRefresh = true)`).
- `MovieViewModelFactory`: Factory sederhana untuk membuat `MovieViewModel` dengan `GetPopularMoviesUseCase`.

/ui/AppNavigation.kt

Mengkonfigurasi navigasi dalam aplikasi menggunakan Jetpack Navigation Compose.

- `rememberNavController()`: Membuat dan mengingat `NavController`.
- `NavHost`: Container untuk tujuan navigasi. `startDestination = "movieList"` menentukan layar awal.
- `composable("movieList") { ... }`: Mendefinisikan rute untuk `MovieListScreen`.
- `composable(route = "movieDetail/{movieId}", arguments = listOf(navArgument("movieId") { type = NavType.StringType }))) { ... }`: Mendefinisikan rute untuk `MovieDetailScreen`. `{movieId}` adalah argumen yang

akan diteruskan. `MovieDetailViewModel` akan mengambilnya menggunakan `SavedStateHandle`.

/ui/Routes.kt

File ini mendefinisikan sebuah object bernama `Routes`. Dalam Kotlin, object adalah cara untuk mendeklarasikan sebuah Singleton, yang berarti hanya akan ada satu instance dari kelas `Routes` di seluruh aplikasi. Pendekatan ini sangat ideal untuk mengelola konstanta dan utilitas yang bersifat global, seperti definisi rute navigasi.

Tujuan utama dari file ini adalah untuk **sentralisasi dan standarisasi rute navigasi** yang digunakan oleh Jetpack Navigation. Ini adalah praktik terbaik yang sangat dianjurkan untuk menghindari apa yang disebut sebagai *"magic strings"* (string literal yang tersebar di banyak file kode).

Di dalam object `Routes`, terdapat beberapa komponen:

- `const val MOVIE_LIST = "movieList"`: Mendefinisikan rute untuk layar daftar film sebagai sebuah konstanta. Dengan menggunakan `Routes.MOVIE_LIST` di seluruh aplikasi (misalnya di `AppNavigation.kt`), Anda memastikan konsistensi dan mengurangi risiko kesalahan ketik. Jika suatu saat Anda perlu mengubah nama rute ini, Anda hanya perlu mengubahnya di satu tempat ini.
- `const val MOVIE_DETAIL = "movieDetail/{movieId}"`: Mendefinisikan rute untuk layar detail film. Bagian `{movieId}` adalah *placeholder* untuk argumen yang akan dikirimkan, dalam hal ini adalah ID dari film yang akan ditampilkan. Mendefinisikan struktur rute ini sebagai konstanta juga memastikan konsistensi.
- `fun navigateById(id: Int): String`: Ini adalah fungsi utilitas yang sangat berguna. Fungsi ini mengambil sebuah id film sebagai parameter `Int` dan mengembalikan `String` rute yang sudah lengkap dan siap digunakan untuk navigasi.

/util/AppSettings.kt

File ini menyediakan utilitas untuk bekerja dengan `SharedPreferences`, yang cocok untuk menyimpan data preferensi atau pengaturan sederhana.

- `PREFS_NAME`: Nama file `SharedPreferences`.

- `getPrefs(context: Context)`: Fungsi helper privat untuk mendapatkan instance `SharedPreferences`.
- `getLastMoviesCacheTimestamp` dan `setLastMoviesCacheTimestamp`: Metode untuk menyimpan dan mengambil timestamp cache film. Perlu dicatat bahwa dalam `MovieRepositoryImpl` yang Anda berikan, timestamp untuk caching data film sebenarnya diambil dari dan disimpan di dalam entitas `MovieEntity` di Room (`lastRefreshed`). Fungsi di `AppSettings` ini mungkin dimaksudkan untuk skenario caching yang berbeda atau sebagai alternatif, tetapi tidak secara aktif digunakan oleh logika caching `MovieRepositoryImpl` untuk film.
- `getTheme` dan `setTheme`: Contoh bagaimana `SharedPreferences` dapat digunakan untuk menyimpan pengaturan tema UI (misalnya "system", "light", "dark").

/util/NetworkResult.kt

Ini adalah implementasi dari **generic response** untuk menangani status dan hasil dari operasi jaringan (atau operasi asinkron lainnya yang bisa gagal).

- `sealed class NetworkResult<T>(val data: T? = null, val message: String? = null)`: Kelas dasar dengan properti untuk data (jika sukses) dan pesan (jika error).
- `class Success<T>(data: T) : NetworkResult<T>(data)`: Merepresentasikan operasi yang sukses, membawa data hasil.
- `class Error<T>(message: String, data: T? = null) : NetworkResult<T>(data, message)`: Merepresentasikan operasi yang gagal, membawa pesan error dan opsional data lama/parsial (berguna untuk menampilkan data cache saat jaringan gagal).

/MainActivity.kt

Ini adalah *Activity* utama dan titik masuk aplikasi Anda.

- `setContent { ... }`: Mengatur konten UI menggunakan Jetpack Compose.
- `ApiComposeTheme { ... }`: Menerapkan tema kustom aplikasi Anda.

- `Surface { AppNavigation() }`: `Surface` adalah container dasar di `Compose`, dan `AppNavigation()` dipanggil untuk mengatur dan menampilkan layar awal (dan navigasi selanjutnya).

/MyApplication.kt

Kelas `MyApplication` yang mewarisi `android.app.Application`.

- `val database: AppDatabase by lazy { AppDatabase.getDatabase(this) }`: Membuat instance `AppDatabase` secara *lazy* dan menyediakannya sebagai properti publik. Ini adalah cara sederhana untuk menyediakan instance database secara global di aplikasi. Instance ini kemudian diakses oleh `ViewModel factories`

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

<https://github.com/raihan2030/Praktikum-Pemrograman-Mobile>