

**LAPORAN PRAKTIKUM  
PEMROGRAMAN MOBILE  
MODUL 4**



**ViewModel and Debugging**

**Oleh:**

**Aliya Raffa Naura Ayu**

**NIM. 2310817120014**

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

**LEMBAR PENGESAHAN**  
**LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE**  
**MODUL 4**

Laporan Praktikum Pemrograman Mobile Modul 4: ViewModel and Debugging ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Aliya Raffa Naura Ayu  
NIM : 2310817120014

Menyetujui,  
Asisten Praktikum

Mengetahui,  
Dosen Penanggung Jawab Praktikum

Zulfa Auliya Akbar  
NIM. 2210817210026

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

## **DAFTAR ISI**

LEMBAR PENGESAHAN .....	2
DAFTAR ISI .....	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL .....	5
SOAL 1.....	6
A. Source Code.....	6
B. Output Program .....	30
C. Pembahasan .....	33
D. Tautan Git.....	47
SOAL 2.....	48
A. Jawaban .....	48

## **DAFTAR GAMBAR**

Gambar 1. Screenshot Output Tampilan Halaman List Soal 1.....	30
Gambar 2. Screenshot Output Tampilan Halaman Detail Soal 1 .....	31
Gambar 3. Screenshot Output Tampilan dari Link Letterboxd.....	31
Gambar 4. Penggunaan Debugger Resume Program (F9) .....	32
Gambar 5. Penggunaan Debugger Step Into (F7).....	32
Gambar 6. Penggunaan Debugger Step Over (F8) .....	33
Gambar 7. Penggunaan Debugger Step Out (Shift+F8) .....	33

## **DAFTAR TABEL**

Tabel 1. Source Code Jawaban Soal 1 MainActivity.kt .....	6
Tabel 2. Source Code Jawaban Soal 1 MovieData.kt.....	8
Tabel 3. Source Code Jawaban Soal 1 movieList.kt .....	9
Tabel 4. Source Code Jawaban Soal 1 glideimage.kt.....	16
Tabel 5. Source Code Jawaban Soal 1 Color.kt .....	17
Tabel 6. Source Code Jawaban Soal 1 MovieListTheme.kt.....	17
Tabel 7. Source Code Jawaban Soal 1 Theme.kt .....	18
Tabel 8. Source Code Jawaban Soal 1 Type.kt .....	19
Tabel 9. Source Code Jawaban Soal 1 MovieDetail.kt .....	20
Tabel 10. Source Code Jawaban Soal 1 MovieList.kt.....	21
Tabel 12. Source Code Jawaban Soal 1 build.gradle.kts.....	24
Tabel 13. Source Code Jawaban Soal 1 MovieRepository.kt.....	27
Tabel 14. Source Code Jawaban Soal 1 MovieViewModel.kt.....	27
Tabel 15. Source Code Jawaban Soal 1 MovieViewModelFactory.kt.....	29

## SOAL 1

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:
  - a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
  - b. Gunakan ViewModelFactory dalam pembuatan ViewModel
  - c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
  - d. gunakan logging untuk event berikut:
    - a. Log saat data item masuk ke dalam list
    - b. Log saat tombol Detail dan tombol Explicit Intent ditekan
    - c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
    - d. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out
2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

### A. Source Code

#### 1. MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1 MainActivity.kt

```
1 package com.example.moviescrollablelist_viewmodel
2
3 import android.os.Bundle
4 import android.util.Log
5 import androidx.activity.ComponentActivity
6 import androidx.activity.compose.setContent
7 import androidx.compose.runtime.LaunchedEffect
8 import androidx.lifecycle.viewmodel.compose.viewModel
```

```

9 import androidx.navigation.NavType
10 import androidx.navigation.compose.NavHost
11 import androidx.navigation.compose.composable
12 import androidx.navigation.compose.rememberNavController
13 import androidx.navigation.navArgument
14 import com.example.moviescrollablelist_viewmodel.data.MovieViewModel
15 import
16 com.example.moviescrollablelist_viewmodel.data.viewModelFactory
17 import com.example.moviescrollablelist_viewmodel.ui.MovieDetail
18 import com.example.moviescrollablelist_viewmodel.ui.MovieList
19 import
20 com.example.moviescrollablelist_viewmodel.ui.theme.MovieListTheme
21
22 class MainActivity : ComponentActivity() {
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         setContent {
26             MovieListTheme {
27                 val navController = rememberNavController()
28                 val movieViewModel: MovieViewModel =
29                     viewModel(factory = viewModelFactory())
30
31                 LaunchedEffect(Unit) {
32                     Log.d("MainActivity", "MainActivity loaded")
33                 }
34
35                 NavHost(
36                     navController = navController,
37                     startDestination = "movieList"
38                 ) {
39                     composable("movieList") {
40                         MovieList(navController = navController)
41                     }
42                     composable(

```

```

43             "movieDetail/{desc}/{image}",
44             arguments = listOf(
45                 navArgument("desc") { type = NavType.StringType },
46                 navArgument("image") { type = NavType.IntType }
47             )
48         ) { backStackEntry ->
49             val desc = backStackEntry.arguments?.getString("desc") ?: ""
50             val image = backStackEntry.arguments?.getInt("image") ?: 0
51             MovieDetail(navController = navController,
52             desc = desc, image = image)
53         }
54     }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
```

## 2. MovieData.kt

Tabel 2. Source Code Jawaban Soal 1 MovieData.kt

```

1 package com.example.moviescrollablelist_viewmodel.data
2
3 data class MovieData(
4     val name: String,
5     val image: Int,
6     val url: String,
7     val description: String,
8 )
```

## 3. movieList.kt

Tabel 3. Source Code Jawaban Soal 1 movieList.kt

```
1 package com.example.moviescrollablelist_viewmodel.data
2
3 import com.example.moviescrollablelist_viewmodel.R
4
5 val movieList = listOf(
6     MovieData(
7         id = 1,
8         name = "Ada Apa Dengan Cinta?",
9         image = R.drawable.aadc,
10        url = "https://letterboxd.com/film/whats-up-with-
11 cinta/",
12        description = "Sinopsis: Ada Apa dengan Cinta? adalah
13 film drama remaja Indonesia yang mengisahkan Cinta, siswi SMA
14 populer yang hobi menulis puisi. Hidupnya berubah saat ia jatuh
15 cinta pada Rangga, siswa pendiam dari latar belakang berbeda.
16 Hubungan mereka yang tumbuh diam-diam membuat Cinta berkonflik
17 dengan sahabat-sahabatnya, memaksanya memilih antara
18 persahabatan atau cinta. Film ini menjadi ikon kisah cinta
19 remaja Indonesia dengan sentuhan puisi dan emosi yang mendalam.
20 Dibalut dengan alunan musik yang ikonik dan puisi-puisi yang
21 menyentuh hati. Ada Apa dengan Cinta? menjadi simbol romantika
22 masa muda yang membekas hingga kini."
23 ),
24     MovieData(
25         id = 2,
26         name = "Gie",
27         image = R.drawable.gie,
28         url = "https://letterboxd.com/film/gie/",
29         description = "Sinopsis: GIE adalah film biografi
30 tentang Soe Hok Gie, aktivis dan penulis idealis yang lantang
31 menyuarakan kebenaran di tengah gejolak politik era Soekarno
32 dan Soeharto. Sebagai mahasiswa yang kritis, Gie menolak segala
33 bentuk penindasan dan korupsi, meski harus berjuang seorang
```

```

34 diri. Film ini menggambarkan perjuangan moral, pencarian makna
35 hidup, dan semangat muda yang tak gentar melawan ketidakadilan.
36 Dengan latar sejarah Indonesia yang penuh gejolak, GIE menjadi
37 potret reflektif tentang suara kritis generasi muda, pentingnya
38 integritas moral, dan keberanian untuk berdiri sendiri dalam
39 memperjuangkan kebenaran, bahkan ketika itu berarti melawan
40 arus."
41 ),
42     MovieData(
43         id = 3,
44         name = "Janji Joni",
45         image = R.drawable.janji_joni,
46         url = "https://letterboxd.com/film/jonis-promise/",
47         description = "Sinopsis: Janji Joni mengisahkan Joni,
48 seorang pengantar gulungan film yang berjuang menepati janji
49 kepada gadis impiannya untuk mengantar film tepat waktu. Namun,
50 di tengah kota yang penuh kekacauan dan kejadian tak terduga,
51 misi sederhana itu berubah menjadi petualangan kocak dan penuh
52 tantangan. Dengan gaya penceritaan yang segar, musik yang
53 catchy, dan nuansa urban yang kuat, Janji Joni menyuguhkan kisah
54 tentang ketekunan, integritas, dan usaha seorang pria biasa yang
55 ingin menepati janji demi cinta yang tumbuh dalam sekejap."
56 ),
57     MovieData(
58         id = 4,
59         name = "Catatan Akhir Sekolah",
60         image = R.drawable.cas,
61         url = "https://letterboxd.com/film/azzamine/",
62         description = "Sinopsis: Kisah tiga sahabat di SMA Fajar
63 Harapan: Agni, Alde, dan Arian. Di sekolah mereka dianggap
64 pecundang dan tak diperhitungkan di sekolah. Karena sifat itu
65 pula Agni kehilangan pacarnya, Alina yang lebih memilih Ray,
66 jagoan sekolah. Tiga orang itu bersahabat karena menjadi bulan-
67 bulanan panitia orientasi pelajar saat jadi murid baru. Tiga

```

```

68 sahabat tadi kemudian berambisi membuktikan kemampuan mereka.
69 Agni yang berambisi menjadi sutradara film, lalu berniat membuat
70 film dokumenter tentang sekolahnya yang ingin diputar pada pesta
71 akhir tahun sekolah. Dibantu sahabatnya yang pandai menulis dan
72 pandai main gitar bas, mereka membuat film dokumenter tentang
73 dinamika kehidupan sekolah: ada cewek yang menyadari dia berada
74 di bawah kekuasaan lelaki, ada kepala sekolah yang ditelanjangi
75 kebejatannya."
76 ),
77 MovieData(
78     id = 5,
79     name = "Marmut Merah Jambu",
80     image = R.drawable.marmut_merah_jambu,
81     url = "https://letterboxd.com/film/pink-guinea-
82 pig/details/",
83     description = "Sinopsis: Marmut Merah Jambu
84 menceritakan Dika, siswa SMA canggung yang diam-diam menyukai
85 gadis bernama Ina. Untuk menarik perhatiannya, ia membentuk Trio
86 Detektif bersama dua sahabatnya dan mulai menyelidiki kasus
87 misterius \"Demonic Pink Guinea Pig\" yang menghebohkan
88 sekolah. Di tengah penyelidikan yang penuh kekonyolan dan
89 kejadian absurd, Dika justru menemukan makna persahabatan,
90 penerimaan diri, dan cinta remaja yang tak selalu berjalan
91 sesuai harapan. Film ini menyuguhkan kisah cinta pertama dengan
92 humor khas Raditya Dika yang segar dan menghibur. "
93 ),
94 MovieData(
95     id = 6,
96     name = "Aruna dan Lidahnya",
97     image = R.drawable.aruna,
98     url = "https://letterboxd.com/film/aruna-her-palate/",
99     description = "Sinopsis: Aruna dan Lidahnya mengisahkan
100 Aruna, seorang epidemiolog yang gemar makan, saat ia melakukan
101 perjalanan ke berbagai daerah untuk menyelidiki kasus flu

```

```

102 burung. Bersama dua sahabatnya—seorang koki dan kritikus
103 makanan—perjalanan dinas itu berubah menjadi petualangan
104 kuliner yang penuh rasa, tawa, dan konflik. Di tengah sajian
105 makanan khas Nusantara, Aruna mulai menyadari makna cinta,
106 persahabatan, dan pencarian jati diri. Lewat visual yang
107 menggoda selera dan dialog yang cerdas, Aruna dan Lidahnya
108 menyuguhkan lebih dari sekadar kisah cinta dan makanan—ini
109 adalah perjalanan emosional yang mempertemukan rasa, memori,
110 dan hati. "
111 ),
112     MovieData(
113         id = 7,
114         name = "Ali dan Ratu Ratu Queens",
115         image = R.drawable.queens,
116         url = "https://letterboxd.com/film/ali-ratu-ratu-
117 queens/",
118         description = "Sinopsis: Ali dan Ratu-Ratu Queens
119 mengikuti perjalanan seorang remaja bernama Ali yang, setelah
120 kehilangan ayahnya, pergi ke New York untuk mencari ibu
121 kandungnya yang telah lama terpisah darinya. Di kota besar itu,
122 Ali bertemu dengan sekelompok wanita Indonesia yang memberinya
123 dukungan dan rasa kebersamaan. Di tengah pencarinya, Ali
124 menemukan cinta pertama dan membangun hubungan yang mengubah
125 hidupnya. Sebuah kisah tentang keluarga, cinta, dan pencarian
126 jati diri yang penuh kehangatan. "
127 ),
128     MovieData(
129         id = 8,
130         name = "Mencuri Raden Saleh",
131         image = R.drawable.raden_saleh,
132         url = "https://letterboxd.com/film/stealing-raden-
133 saleh/",
134         description = "Sinopsis: Mencuri Raden Saleh
135 mengisahkan seorang ahli pembuat lukisan palsu yang

```

```

136 merencanakan pencurian terbesar untuk menyelamatkan ayahnya. Ia
137 membentuk tim yang terdiri dari para ahli untuk mencuri lukisan
138 Raden Saleh yang sangat berharga. Dalam perjalanan yang penuh
139 ketegangan dan rintangan, mereka menghadapi tantangan dari
140 pihak berwenang dan musuh-musuh yang siap menggagalkan misi
141 mereka. Sebuah film aksi thriller yang menggali tema
142 persahabatan, pengorbanan, dan moralitas, dengan alur cerdas
143 dan penuh kejutan."
144 ),
145     MovieData(
146         id = 9,
147         name = "The Big 4",
148         image = R.drawable.big_4,
149         url = "https://letterboxd.com/film/the-big-4/",
150         description = "Sinopsis: The Big 4 mengisahkan Dina,
151 seorang detektif perempuan yang teguh mengikuti aturan, namun
152 harus bergabung dengan empat pembunuh bayaran yang eksentrik
153 dan sedang terpuruk dalam hidup mereka. Ketika ayah Nina dibunuh
154 dengan cara misterius, ia terpaksa bekerja sama dengan tim yang
155 kacau ini untuk mengungkap kebenaran. Dalam penyelidikan yang
156 penuh kekacauan, aksi, dan komedi, mereka mulai menggali
157 konspirasi besar yang melibatkan dunia kriminal yang berbahaya.
158 Meskipun sering bertentangan, Dina dan timnya belajar untuk
159 saling melengkapi dan bekerja sama, membentuk ikatan yang tak
160 terduga dalam upaya mereka mencari keadilan."
161 ),
162     MovieData(
163         id = 10,
164         name = "A+",
165         image = R.drawable.aplus,
166         url = "https://letterboxd.com/film/a-2023/",
167         description = "Sinopsis: Berjuang dari barisan try out
168 untuk menembus peringkat pertama, Kaliypto Dirgantari harus
169 menghadapi empat besar pemegang tahta di SMA Bina Indonesia: Re

```

170	Dirgantara, Kenan Aditya, Adinda Aletheia dan Aurora Calista.
171	Masing-masing didorong oleh motivasi mereka sendiri, kelimanya
172	bersaing ketat untuk mendapatkan peringkat paralel, setidaknya
173	sampai rahasia epik terkait sistem sekolah terungkap."
174	) ,
175	MovieData(
176	id = 11,
177	name = "Gadis Kretek",
178	image = R.drawable.gadis_kretek,
179	url = "https://letterboxd.com/film/cigarette-girl-
180	2023/",
181	description = "Sinopsis: Soeraja merupakan pemilik
182	pabrik kretek Djagad Raya yang sedang sekarat. Namun, ia justru
183	ingin bertemu perempuan yang bukanistrinya, yakni Jeng Yah.
184	Sang istri pun cemburu karena hal tersebut merupakan permintaan
185	terakhir suaminya. Akan tetapi, ketiga anak Soeraja, yakni
186	Lebas, Karim, dan Tegar tetap berusaha mencari keberadaan Jeng
187	Yah ke pelosok Pulau Jawa. Ketika berada dalam perjalanan untuk
188	mencari jejak Jeng Yah, mereka bertemu buruh batil yang menguak
189	asal-usul Kretek Djagad Raya hingga menjadi kretek nomor satu
190	di Indonesia dan juga mengetahui kisah cinta sang ayah dengan
191	Jeng Yah."
192	) ,
193	MovieData(
194	id = 12,
195	name = "Home Sweet Loan",
196	image = R.drawable.home_sweet_loan,
197	url = "https://letterboxd.com/film/home-sweet-loan/",
198	description = "Sinopsis: Home Sweet Loan mengisahkan
199	perjuangan Kaluna, seorang wanita muda yang bekerja di bank
200	dengan gaji pas-pasan. Kaluna bermimpi memiliki rumah sendiri
201	di Jakarta, namun menghadapi berbagai tantangan finansial. Ia
202	tinggal bersama keluarga besarnya yang terdiri dari orang tua,
203	dua kakak beserta keluarga mereka dalam satu rumah yang sempit.

```

204 Bersama tiga sahabatnya Tanisha, Kamamiya, dan Danan - Kaluna
205 berusaha keras mencari hunian terjangkau di pinggiran Jakarta.
206 Ia harus memilih antara memperjuangkan mimpiya sendiri atau
207 membantu keluarganya yang terlilit masalah utang. Film ini
208 menggambarkan realita generasi sandwich dalam menghadapi
209 tekanan finansial dan keluarga. Mereka rela mengerem
210 pengeluaran dan mencari kerja sampingan demi mewujudkan impian
211 tersebut. Namun, Kaluna dihadapkan pada dilema saat kondisi
212 keuangan keluarganya memburuk dan membutuhkan bantuan. "
213 ),
214     MovieData(
215         id = 13,
216         name = "Dua Hati Biru",
217         image = R.drawable.dua_hati_biru,
218         url = "https://letterboxd.com/film/home-sweet-loan/",
219         description = "Sinopsis: Kisah Bima dan Dara berlanjut.
220 Pasangan suami-istri muda yang dulu menikah karena sebuah
221 kesalahan. Empat tahun berlalu, Dara memutuskan kembali ke
222 Jakarta demi berkumpul dengan keluarga kecilnya, Bima dan Adam.
223 Berpisah sejak Adam lahir, Dara harus berusaha mendekatkan diri
224 dengan sang buah hati meskipun terdapat berbagai masalah dalam
225 rumah tangganya. "
226 ),
227     MovieData(
228         id = 14,
229         name = "Jumbo",
230         image = R.drawable.jumbo,
231         url = "https://letterboxd.com/film/jumbo-2025/",
232         description = "Sinopsis: Jumbo bercerita tentang Don,
233 seorang anak yatim piatu berusia 10 tahun dengan tubuh besar
234 yang sering diremehkan. Don menemukan pelarian dan inspirasi
235 dalam buku dongeng warisan orang tuanya, yang penuh cerita
236 ajaib. Untuk membuktikan dirinya, Don bertekad mengikuti
237 pertunjukan bakat dengan sandiwara yang terinspirasi dari buku

```

238	tersebut. Namun, mimpi Don dihina oleh teman-temannya, dan buku
239	dongengnya dicuri oleh perundung. Untungnya, Don mendapat
240	dukungan dari Oma dan sahabat-sahabatnya. Dalam usahanya
241	mendapatkan kembali bukunya, Don bertemu dengan Meri, seorang
242	anak dari dunia lain yang butuh bantuan untuk menemukan orang
243	tuanya. Petualangan penuh keajaiban pun dimulai, mengajarkan
244	arti persahabatan, keberanian, dan kepercayaan diri. "
245	) ,
246	)

#### 4. glideimage.kt

Tabel 4. Source Code Jawaban Soal 1 glideimage.kt

1	package com.example.moviescrollablelist_viewmodel.ui.components
2	
3	import android.widget.ImageView
4	import androidx.annotation.DrawableRes
5	import androidx.compose.runtime.Composable
6	import androidx.compose.ui.Modifier
7	import androidx.compose.ui.viewinterop.AndroidView
8	import com.bumptech.glide.Glide
9	
10	@Composable
11	fun GlideImage(
12	@DrawableRes resId: Int,
13	contentDescription: String?,
14	modifier: Modifier = Modifier
15	) {
16	AndroidView(
17	factory = { context ->
18	ImageView(context).apply {
19	scaleType = ImageView.ScaleType.FIT_CENTER
20	contentDescription?.let {
21	this.contentDescription = it }
22	}

```

23     },
24     update = { imageView ->
25         Glide.with(imageView.context)
26             .load(resId)
27             .into(imageView)
28     },
29     modifier = modifier
30 )
31 }
```

## 5. Color.kt

Tabel 5. Source Code Jawaban Soal 1 Color.kt

```

1 package com.example.moviescrollablelist_viewmodel.ui.theme
2 import androidx.compose.ui.graphics.Color
3
4 val Purple80 = Color(0xFFD0BCFF)
5 val PurpleGrey80 = Color(0xFFCCC2DC)
6 val Pink80 = Color(0xFFEFB8C8)
7
8 val Purple40 = Color(0xFF6650a4)
9 val PurpleGrey40 = Color(0xFF625b71)
10 val Pink40 = Color(0xFF7D5260)
```

## 6. MovieListTheme.kt

Tabel 6. Source Code Jawaban Soal 1 MovieListTheme.kt

```

1 package com.example.moviescrollablelist_viewmodel.ui.theme
2
3 import androidx.compose.material3.MaterialTheme
4 import androidx.compose.material3.darkColorScheme
5 import androidx.compose.material3.lightColorScheme
6 import androidx.compose.runtime.Composable
7
8 private val DarkColorScheme = darkColorScheme()
```

```

9 private val LightColorScheme = lightColorScheme()
10
11 @Composable
12 fun MovieListTheme(content: @Composable () -> Unit) {
13     MaterialTheme(
14         colorScheme = LightColorScheme,
15         typography = Typography,
16         content = content
17     )
18 }
```

## 7. Theme.kt

Tabel 7. Source Code Jawaban Soal 1 Theme.kt

```

1 package com.example.moviesscrollablelist_viewmodel.ui.theme
2
3 import androidx.compose.material3.MaterialTheme
4 import androidx.compose.material3.darkColorScheme
5 import androidx.compose.material3.lightColorScheme
6 import androidx.compose.runtime.Composable
7 import androidx.compose.ui.graphics.Color
8
9 private val LightColors = lightColorScheme(
10     primary = Color(0xFF6200EE),
11     secondary = Color(0xFF03DAC6),
12 )
13
14 private val DarkColors = darkColorScheme(
15     primary = Color(0xFFBB86FC),
16     secondary = Color(0xFF03DAC6),
17 )
18
19 @Composable
20 fun MovieListTheme(
21     darkTheme: Boolean = false,
```

```

22     content: @Composable () -> Unit
23 ) {
24     val colors = if (darkTheme) DarkColors else LightColors
25
26     MaterialTheme (
27         colorScheme = colors,
28         typography = Typography,
29         content = content
30     )
31 }

```

## 8. Type.kt

Tabel 8. Source Code Jawaban Soal 1 Type.kt

```

1 package com.example.moviesscrollablelist_viewmodel.ui.theme
2
3 import androidx.compose.material3.Typography
4 import androidx.compose.ui.text.TextStyle
5 import androidx.compose.ui.text.font.FontFamily
6 import androidx.compose.ui.text.font.FontWeight
7 import androidx.compose.ui.unit.sp
8
9 val Typography = Typography(
10     bodyLarge = TextStyle(
11         fontFamily = FontFamily.Default,
12         fontWeight = FontWeight.Normal,
13         fontSize = 16.sp,
14         lineHeight = 24.sp,
15         letterSpacing = 0.5.sp
16     )
17 )

```

## 9. MovieDetail.kt

Tabel 9. Source Code Jawaban Soal 1 MovieDetail.kt

```
1 package com.example.moviescrollablelist_viewmodel.ui
2
3 import androidx.compose.foundation.Image
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Spacer
6 import androidx.compose.foundation.layout.fillMaxSize
7 import androidx.compose.foundation.layout.fillMaxWidth
8 import androidx.compose.foundation.layout.height
9 import androidx.compose.foundation.layout.padding
10 import androidx.compose.foundation.rememberScrollState
11 import androidx.compose.foundation.verticalScroll
12 import androidx.compose.material.icons(Icons)
13 import androidx.compose.material.icons.filled.ArrowBack
14 import androidx.compose.material3.Icon
15 import androidx.compose.material3.IconButton
16 import androidx.compose.material3.Text
17 import androidx.compose.runtime.Composable
18 import androidx.compose.ui.Modifier
19 import androidx.compose.ui.graphics.painter.Painter
20 import androidx.compose.ui.res.painterResource
21 import androidx.compose.ui.unit.dp
22 import androidx.compose.ui.unit.sp
23 import androidx.navigation.NavController
24
25 @Composable
26 fun MovieDetail(navController: NavController, desc: String,
27 image: Int) {
28     Column(
29         modifier = Modifier
30             .fillMaxSize()
31             .verticalScroll(rememberScrollState())
32             .padding(25.dp)
33     ) {
```

```

34        IconButton(onClick = { navController.popBackStack() })
35    {
36        Icon(imageVector = Icons.Filled.ArrowBack,
37        contentDescription = "Back")
38    }
39
40    ImageGlide(
41        painter = painterResource(id = image),
42        contentDescription = null,
43        modifier = Modifier
44            .fillMaxWidth()
45            .height(300.dp)
46    )
47
48    Spacer(modifier = Modifier.height(16.dp))
49    Text(text = desc, fontSize = 18.sp)
50 }
51 }
52
53 @Composable
54 fun ImageGlide(painter: Painter, contentDescription: Nothing?,
55 modifier: Modifier) {
56     Image(painter = painter, contentDescription =
57     contentDescription, modifier = modifier)
58 }
59

```

## 10. MovieList.kt

Tabel 10. Source Code Jawaban Soal 1 MovieList.kt

```

1 package com.example.moviescrollablelist_viewmodel.ui
2
3 import android.content.Intent
4 import android.net.Uri

```

```
5 import androidx.compose.foundation.Image
6 import androidx.compose.foundation.layout.Column
7 import androidx.compose.foundation.layout.Row
8 import androidx.compose.foundation.layout.Spacer
9 import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.height
11 import androidx.compose.foundation.layout.padding
12 import androidx.compose.foundation.layout.width
13 import androidx.compose.foundation.lazy.LazyColumn
14 import androidx.compose.foundation.shape.RoundedCornerShape
15 import androidx.compose.material3.Button
16 import androidx.compose.material3.Card
17 import androidx.compose.material3.CardDefaults
18 import androidx.compose.material3.Text
19 import androidx.compose.runtime.Composable
20 import androidx.compose.ui.Alignment
21 import androidx.compose.ui.Modifier
22 import androidx.compose.ui.platform.LocalContext
23 import androidx.compose.ui.res.painterResource
24 import androidx.compose.ui.text.font.FontWeight
25 import androidx.compose.ui.text.style.TextOverflow
26 import androidx.compose.ui.unit.dp
27 import androidx.compose.ui.unit.sp
28 import androidx.navigation.NavHostController
29 import com.example.moviescrollablelist_viewmodel.data.movieList
30
31 @Composable
32 fun MovieList(navController: NavHostController) {
33     val context = LocalContext.current
34
35     LazyColumn(modifier = Modifier.padding(16.dp)) {
36         items(movieList.size) { index ->
37             val movie = movieList[index]
38             Card(
```

```

39         modifier = Modifier
40             .fillMaxWidth()
41             .padding(vertical = 8.dp),
42             shape = RoundedCornerShape(16.dp),
43             elevation = CardDefaults.cardElevation(4.dp)
44     ) {
45         Row(
46             verticalAlignment =
47 Alignment.CenterVertically,
48             modifier = Modifier.padding(16.dp)
49     ) {
50         Image(
51             painter      = painterResource(id =
52 movie.image),
53             contentDescription = null,
54             modifier = Modifier
55                 .width(100.dp)
56                 .height(140.dp)
57         )
58         Spacer(modifier = Modifier.width(16.dp))
59         Column(modifier = Modifier.weight(1f)) {
60             Text(text = movie.name,    fontSize =
61 20.sp, fontWeight = FontWeight.Bold)
62             Text(
63                 text = movie.description,
64                 fontSize = 14.sp,
65                 maxLines = 3,
66                 overflow = TextOverflow.Ellipsis
67             )
68             Spacer(modifier =
69 Modifier.height(8.dp))
70         Row {
71             Button(
72                 onClick = {

```

```

73                                     val           intent      =
74     Intent(Intent.ACTION_VIEW, Uri.parse(movie.url))
75
76     context.startActivity(intent)
77             }
78         )  {
79             Text("Letterboxed")
80         }
81             Spacer(modifier
82     Modifier.width(8.dp))
83             Button(
84                 onClick = {
85
86     navController.navigate("movieDetail/${Uri.encode(movie.description)}/${movie.image}")
87
88             }
89         )  {
90             Text("Detail")
91         }
92     }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
```

## 11. build.gradle.kts

Tabel 11. Source Code Jawaban Soal 1 build.gradle.kts

1	plugins {
2	alias(libs.plugins.android.application)
3	alias(libs.plugins.kotlin.android)
4	alias(libs.plugins.kotlin.compose)

```
5     id("org.jetbrains.kotlin.kapt")
6 }
7
8 android {
9     namespace = "com.example.moviescrollablelist_viewmodel"
10    compileSdk = 35
11
12    defaultConfig {
13        applicationId =
14 "com.example.moviescrollablelist_viewmodel"
15        minSdk = 30
16        targetSdk = 35
17        versionCode = 1
18        versionName = "1.0"
19
20        testInstrumentationRunner =
21 "androidx.test.runner.AndroidJUnitRunner"
22    }
23
24    buildTypes {
25        release {
26            isMinifyEnabled = false
27            proguardFiles(
28                getDefaultProguardFile("proguard-android-
29 optimize.txt"),
29
30                "proguard-rules.pro"
31            )
32        }
33    }
34
35    compileOptions {
36        sourceCompatibility = JavaVersion.VERSION_11
37        targetCompatibility = JavaVersion.VERSION_11
38    }
```

```
39
40     kotlinOptions {
41         jvmTarget = "11"
42     }
43
44     buildFeatures {
45         compose = true
46     }
47
48     composeOptions {
49         kotlinCompilerExtensionVersion = "1.5.3"
50     }
51 }
52
53 dependencies {
54     implementation(platform("androidx.compose:compose-
55 bom:2023.03.00"))
56
57 androidTestImplementation(platform("androidx.compose:compose-
58 bom:2023.03.00"))
59
60     implementation("com.github.bumptech.glide:glide:4.15.1")
61     kapt("com.github.bumptech.glide:compiler:4.15.1")
62
63     implementation("androidx.compose.ui:ui")
64     implementation("androidx.compose.material3:material3")
65     implementation("androidx.navigation:navigation-
66 compose:2.7.0")
67     implementation("io.coil-kt:coil-compose:2.4.0")
68     implementation("androidx.activity:activity-compose:1.7.2")
69     implementation("androidx.lifecycle:lifecycle-runtime-
70 ktx:2.6.1")
71     implementation("androidx.compose.ui:ui-tooling-preview")
72     debugImplementation("androidx.compose.ui:ui-tooling")
```

```

73
74     testImplementation("junit:junit:4.13.2")
75     androidTestImplementation("androidx.test.ext:junit:1.1.5")
76     androidTestImplementation("androidx.test.espresso:espresso-
77 core:3.5.1")
78     androidTestImplementation("androidx.compose.ui:ui-test-
79 junit4")
80     debugImplementation("androidx.compose.ui:ui-test-manifest")
81 }
```

## 12. MovieRepository.kt

Tabel 12. Source Code Jawaban Soal 1 MovieRepository.kt

```

1 package com.example.moviescrollablelist_viewmodel.data
2
3 import com.example.moviescrollablelist_viewmodel.data.MovieData
4 import com.example.moviescrollablelist_viewmodel.data.movieList
5
6
7 class MovieRepository {
8     fun getMovies(): List<MovieData> {
9         return movieList
10    }
11
12    fun getMovieById(movieId: Int): MovieData? {
13        return movieList.find { it.id == movieId }
14    }
15 }
```

## 13. MovieViewModel.kt

Tabel 13. Source Code Jawaban Soal 1 MovieViewModel.kt

```

1 package com.example.moviescrollablelist_viewmodel.data
2
3 import androidx.lifecycle.ViewModel
```

```
4 import androidx.lifecycle.viewModelScope
5 import
6 com.example.moviescrollablelist_viewmodel.data.MovieRepository
7 import com.example.moviescrollablelist_viewmodel.data.MovieData
8 import kotlinx.coroutines.flow.MutableStateFlow
9 import kotlinx.coroutines.flow.StateFlow
10 import kotlinx.coroutines.launch
11 import android.util.Log
12
13 class MovieViewModel(private val repository: MovieRepository) :
14 ViewModel() {
15
16     private val _movieList = MutableStateFlow<List<MovieData>>(emptyList())
17     val movieList: StateFlow<List<MovieData>> get() = _movieList
18
19     private val _selectedMovie = MutableStateFlow<MovieData?>(null)
20     val selectedMovie: StateFlow<MovieData?> get() = _selectedMovie
21
22     _selectedMovie
23
24
25     init {
26         Log.d("MovieViewModel", "ViewModel created")
27         getMovies()
28     }
29
30     private fun getMovies() {
31         viewModelScope.launch {
32             _movieList.value = repository.getMovies()
33             Log.d("MovieViewModel", "Movies loaded: ${_movieList.value.size}")
34         }
35     }
36
37 }
```

```

38     fun selectMovie(movieId: Int) {
39         viewModelScope.launch {
40             val movie = repository.getMovieById(movieId)
41             _selectedMovie.value = movie
42             Log.d("MovieViewModel", "Selected movie: "
43 ${movie?.name})
44         }
45     }
46 }
```

## 14. MovieViewModelFactory.kt

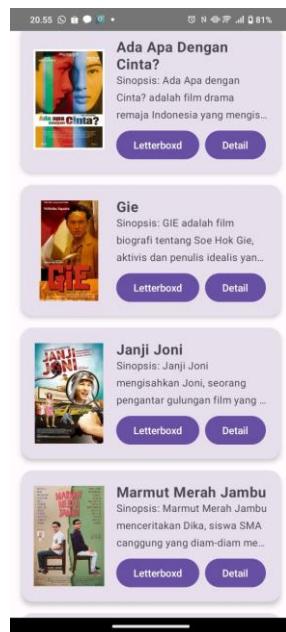
Tabel 14. Source Code Jawaban Soal 1 MovieViewModelFactory.kt

```

1 package com.example.moviescrollablelist_viewmodel.data
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5 import
6 com.example.moviescrollablelist_viewmodel.data.MovieRepository
7
8 @SuppressLint("unchecked")
9 class MovieViewModelFactory(private val repository:
10 MovieRepository) : ViewModelProvider.Factory {
11     override fun <T : ViewModel> create(modelClass: Class<T>):
12 T {
13     if
14 (modelClass.isAssignableFrom(MovieViewModel::class.java)) {
15         return MovieViewModel(repository) as T
16     }
17     throw IllegalArgumentException("Unknown ViewModel
18 class")
19 }
20 }
21
22 fun viewModelFactory(): ViewModelProvider.Factory {
```

```
23     val repository = MovieRepository()  
24     return MovieViewModelFactory(repository)  
25 }
```

## B. Output Program



Gambar 1. Screenshot Output Tampilan Halaman List Soal 1



Sinopsis: Ada Apa dengan Cinta? adalah film drama remaja Indonesia yang mengisahkan Cinta, siswi SMA populer yang hobinya menulis puisi. Hidupnya berubah saat ia jatuh cinta pada Ranga, siswa pendiam dari latar belakang berbeda. Hubungan mereka yang tumbuh diam-diam membuat Cinta berkonflik dengan sahabat-sahabatnya, memaksaanya memilih antara persahabatan atau cinta. Film ini menjadi ikon kisah cinta remaja Indonesia dengan sentuhan puisi dan emosi yang mendalam. Dibalut dengan alunan musik yang ikonik dan puisi-puisi yang menyentuh hati. Ada Apa dengan Cinta? menjadi simbol romantis masa muda yang membekas hingga kini.

Gambar 2. Screenshot Output Tampilan Halaman Detail Soal 1



Gambar 3. Screenshot Output Tampilan dari Link Letterboxd

```
35     LazyColumn(modifier = Modifier.padding(16.dp)) {
36         items(movieList.size) { index -> index: 1
37             val movie = movieList[index] index: 1
38             Card(
39                 modifier = Modifier
40                     .fillMaxWidth()
41                     .padding(vertical = 8.dp),
42                 shape = RoundedCornerShape(16.dp),
43                 elevation = CardDefaults.cardElevation(4.dp)
44             ) {
45                 Row(
46                     verticalAlignment = Alignment.CenterVertically,
47                     modifier = Modifier.padding(16.dp)
48             ) {
49             
```

Debug app x

Threads & Variables Console

✓ "main"@15,76...ain: RUNNING

invoke:37, MovieListKt\$MovieList\$1\$1\$1

214 hidden frames

Switch frames from anywhere in the IDE with Ctrl.. X

MODUL-4 > app > src > main > java > com > example > moviescrollablelist\_viewmodel > ui > MovieList.kt

Gambar 4. Penggunaan Debugger Resume Program (F9)

```
5     val movieList = listOf(
6         MovieData(
7             id = 1,
8             name = "Ada Apa Dengan Cinta?",
9             image = R.drawable.aadc,
10            url = "https://letterboxd.com/film/whats-up-with-cinta/",
11            description = "Sinopsis: Ada Apa dengan Cinta? adalah film drama remaja Indonesia yang mengisahkan Cinta, sis"
12        ),
13        MovieData(
14            id = 2,
15            name = "Gie",
16            image = R.drawable.gie,
17            url = "https://letterboxd.com/film/gie/",
18            description = "Sinopsis: GIE adalah film biografi tentang Soe Hok Gie, aktivis dan penulis idealis yang lanta"
19        ),
20        MovieData(
21        
```

Debug app x

Threads & Variables Console

✓ "main"@15,76...ain: RUNNING

getMovieList:5, MovieListKt (com.example)

invoke:37, MovieListKt\$MovieList\$1\$1\$1

214 hidden frames

Switch frames from anywhere in the IDE with Ctrl.. X

MODUL-4 > app > src > main > java > com > example > moviescrollablelist\_viewmodel > data > movieList.kt

Gambar 5. Penggunaan Debugger Step Into (F7)

```

35     LazyColumn(modifier = Modifier.padding(16.dp)) {
36         items(movieList.size) { index -> index: 1
37             val movie = movieList[index] index: 1 movie: MovieData(id=2, name=Gie, image=https://letterboxd.com/film/gie/, description="")
38             Card(
39                 modifier = Modifier
40                     .fillMaxWidth()
41                     .padding(vertical = 8.dp),
42                     shape = RoundedCornerShape(16.dp),
43                     elevation = CardDefaults.cardElevation(4.dp)
44             ) {
45                 Row(
46                     verticalAlignment = Alignment.CenterVertically,
47                     modifier = Modifier.padding(16.dp)
48             ) {

```

Gambar 6. Penggunaan Debugger Step Over (F8)

```

35     LazyColumn(modifier = Modifier.padding(16.dp)) {
36         items(movieList.size) { index -> index: 2
37             val movie = movieList[index] index: 2
38             Card(
39                 modifier = Modifier
40                     .fillMaxWidth()
41                     .padding(vertical = 8.dp),
42                     shape = RoundedCornerShape(16.dp),
43                     elevation = CardDefaults.cardElevation(4.dp)
44             ) {
45                 Row(
46                     verticalAlignment = Alignment.CenterVertically,
47                     modifier = Modifier.padding(16.dp)
48             ) {

```

Gambar 7. Penggunaan Debugger Step Out (Shift+F8)

## C. Pembahasan

### 1. MainActivity.kt

Pada baris [1], package com.example.moviescrollablelist\_viewmodel menentukan namespace aplikasi. Pada baris [3-20], import berbagai kelas penting seperti Bundle, Log, ComponentActivity, setContent (Compose),

LaunchedEffect, ViewModel, dan komponen navigasi Compose. Pada baris [22], MainActivity mewarisi ComponentActivity sebagai entry point aplikasi. Pada baris [23-24], mengoverride fungsi onCreate dan memanggil superclass-nya. Pada baris [25-26], setContent mulai Compose UI, membungkus dengan tema MovieListTheme. Pada baris [27], navController diinisialisasi untuk mengelola navigasi antar layar. Pada baris [28-29], movieViewModel diinisialisasi dengan factory untuk mengelola data dan state aplikasi. Pada baris [31-33], LaunchedEffect digunakan untuk menjalankan log debug saat Activity pertama kali dibuat. Pada baris [35-49], NavHost mendefinisikan rute navigasi dengan dua layar yaitu rute "movieList" menampilkan daftar film dan rute "movieDetail/{desc}/{image}" menampilkan detail film dengan argumen deskripsi dan gambar. Pada baris [50-57], argumen desc dan image diambil dari backStackEntry dan diteruskan ke MovieDetail.

## 2. MovieData.kt

Pada baris [1], package com.example.moviescrollablelist.data digunakan untuk menentukan namespace atau nama paket dari file ini berada di dalam folder data aplikasi. Pada baris [3], data class MovieData(...) digunakan untuk mendefinisikan data class bernama MovieData yang menyimpan informasi tentang sebuah film. Pada baris [4], val name: String menyatakan properti name bertipe String untuk menyimpan nama atau judul film. Pada baris [5], val image: Int menyatakan properti image bertipe Int yang biasanya mereferensikan ID resource gambar dari film. Pada baris [6], val url: String menyatakan properti url bertipe String yang digunakan untuk menyimpan tautan. Pada baris [7], val description: String menyatakan properti description bertipe String yang menyimpan deskripsi film.

## 3. movieList.kt

Pada baris [1], package com.example.moviescrollablelist.data digunakan untuk menentukan namespace atau nama paket dari file ini agar dapat diatur dan diakses dengan struktur modular di aplikasi. Pada baris [3], import

com.example.moviescrollablelist.R digunakan untuk mengimpor file R yang berisi referensi ke semua resource (seperti gambar drawable) dalam aplikasi. Pada baris [5], val movieList = listOf(...) digunakan untuk mendeklarasikan variabel movieList sebagai daftar berisi beberapa objek MovieData. Pada baris [6], MovieData(...) digunakan untuk membuat sebuah instance dari data class MovieData, yang menyimpan informasi detail dari satu film, seperti nama, gambar, url, dan deskripsi.

Pada baris [7], id = 1 merupakan angka unik yang menjadi identitas film. Pada baris [8], name = "Ada Apa Dengan Cinta?" menetapkan judul film yang ditampilkan. Pada baris [9], image = R.drawable.aadc menetapkan referensi resource gambar drawable untuk film tersebut. Pada baris [10-11], url = "https://letterboxd.com/film/whats-up-with-cinta/" adalah tautan eksternal ke halaman film. Pada baris [12], description = "...", menyimpan deskripsi panjang film dalam bentuk string, menjelaskan sinopsis dan tema utama. Pada baris [24] dan seterusnya, proses pengisian daftar movieList dilanjutkan dengan MovieData(...) berikutnya, masing-masing menyimpan data untuk film lain seperti Gie, Janji Joni, Marmut Merah Jambu, dan seterusnya. Pada baris terakhir (sekitar baris [246]), tanda kurung tutup ) digunakan untuk menutup pemanggilan fungsi listOf yang mencakup seluruh daftar film.

#### 4. glideimage.kt

Pada baris [1], package com.example.moviescrollablelist.ui.components digunakan untuk menentukan namespace atau nama paket tempat komponen UI ini berada dalam proyek. Pada baris [3], import android.widget.ImageView digunakan untuk mengimpor class ImageView dari Android SDK yang akan digunakan untuk menampilkan gambar. Pada baris [4], import androidx.annotation.DrawableRes digunakan untuk memberi anotasi pada parameter agar hanya menerima ID dari resource drawable. Pada baris [5], import

`androidx.compose.runtime.Composable` digunakan untuk menandai fungsi `GlideImage` sebagai fungsi composable dalam Jetpack Compose.

Pada baris [6], `import androidx.compose.ui.Modifier` digunakan untuk memungkinkan penerapan modifikasi tampilan seperti ukuran, padding, dll., pada composable. Pada baris [7], `import androidx.compose.ui.viewinterop.AndroidView` digunakan untuk menyisipkan view tradisional Android (`ImageView`) ke dalam Compose UI.

Pada baris [8], `import com.bumptech.glide.Glide` digunakan untuk mengimpor library Glide yang digunakan untuk memuat gambar secara efisien ke dalam `ImageView`. Pada baris [10], `@Composable` digunakan untuk menandai fungsi `GlideImage` agar bisa digunakan sebagai bagian dari UI deklaratif di Jetpack Compose. Pada baris [11], `fun GlideImage(...)` menetapkan nama fungsi `GlideImage` yang menerima resource ID gambar, deskripsi konten, dan modifier untuk tampilan. Pada baris [12], `@DrawableRes resourceId: Int` menetapkan bahwa parameter `resourceId` harus berupa ID dari resource drawable. Pada baris [13], `contentDescription: String?` merupakan teks alternatif untuk aksesibilitas atau pembaca layar.

Pada baris [14], `modifier: Modifier = Modifier` menetapkan parameter opsional untuk styling komponen. Pada baris [16], `AndroidView(...)` digunakan untuk menyisipkan komponen view Android tradisional ke dalam UI Compose. Pada baris [17], `factory = { context ->` digunakan untuk membuat instance baru dari `ImageView` menggunakan `context`. Pada baris [18], `ImageView(context).apply { ... }` membuat objek `ImageView` dan langsung menetapkan properti-propertinya menggunakan `apply`. Pada baris [19], `scaleType = ImageView.ScaleType.FIT_CENTER` mengatur tampilan gambar agar di-scale dan diposisikan di tengah.

Pada baris [21-22], `contentDescription?.let { ... }` menetapkan deskripsi konten ke `ImageView` jika tersedia, demi mendukung aksesibilitas. Pada baris [25], `update = { imageView -> ... }` berfungsi memperbarui konten view dengan resource gambar menggunakan Glide saat composable dirender ulang. Pada baris [26],

Glide.with(imageView.context) memulai proses pemuatan gambar menggunakan context dari ImageView. Pada baris [27], .load(resId) menetapkan resource drawable yang akan dimuat oleh Glide. Pada baris [28], .into(imageView) menetapkan target ImageView tempat gambar akan ditampilkan. Pada baris [30], modifier = modifier menetapkan modifier Compose yang dikirim dari luar ke AndroidView untuk styling tampilan. Pada baris [28], ) menutup pemanggilan fungsi AndroidView.

## 5. Color.kt

Pada baris [1], package com.example.moviescrollablelist.ui.theme digunakan untuk menentukan namespace atau nama paket dari file tema aplikasi ini. Pada baris [3], import androidx.compose.ui.graphics.Color digunakan untuk mengimpor class Color dari Jetpack Compose yang digunakan untuk mendefinisikan warna-warna kustom. Pada baris [5], val Purple80 = Color(0xFFD0BCFF) mendefinisikan warna ungu muda dengan kode heksadesimal D0BCFF dan menyimpannya dalam variabel bernama Purple80. Pada baris [6], val PurpleGrey80 = Color(0xFFCCC2DC) mendefinisikan warna abu-abu keunguan terang dan menyimpannya dalam variabel PurpleGrey80.

Pada baris [7], val Pink80 = Color(0xFFFFFB8C8) mendefinisikan warna merah muda terang dan menyimpannya dalam variabel Pink80. Pada baris [9], val Purple40 = Color(0xFF6650a4) mendefinisikan warna ungu tua dan menyimpannya dalam variabel Purple40. Pada baris [10], val PurpleGrey40 = Color(0xFF625b71) mendefinisikan warna abu-abu keunguan tua dan menyimpannya dalam variabel PurpleGrey40. Pada baris [11], val Pink40 = Color(0xFF7D5260) mendefinisikan warna merah muda tua dan menyimpannya dalam variabel Pink40.

## 6. MovieListTheme.kt

Pada baris [1], package com.example.moviescrollablelist.ui.theme digunakan untuk menentukan namespace atau nama paket dari file tema dalam aplikasi ini. Pada baris [3], import

`androidx.compose.material3.MaterialTheme` digunakan untuk mengimpor `MaterialTheme` dari Material 3, yang menjadi dasar styling UI. Pada baris [4], `import androidx.compose.material3.darkColorScheme` digunakan untuk mengimpor fungsi yang menghasilkan skema warna tema gelap. Pada baris [5], `import androidx.compose.material3.lightColorScheme` digunakan untuk mengimpor fungsi yang menghasilkan skema warna tema terang.

Pada baris [6], `import androidx.compose.runtime.Composable` digunakan untuk menandai fungsi sebagai komposable, artinya fungsi tersebut bisa digunakan di dalam UI deklaratif Jetpack Compose. Pada baris [8], `private val DarkColorScheme = darkColorScheme()` mendeklarasikan variabel privat bernama `DarkColorScheme` yang menyimpan skema warna default untuk tema gelap. Pada baris [9], `private val LightColorScheme = lightColorScheme()` mendeklarasikan variabel privat bernama `LightColorScheme` yang menyimpan skema warna default untuk tema terang. Pada baris [11], `@Composable` menandai fungsi `MovieListTheme` sebagai fungsi komposable yang bisa digunakan dalam struktur UI Compose. Pada baris [12], `fun MovieListTheme(content: @Composable () -> Unit)` mendeklarasikan fungsi tema yang menerima parameter `content` berupa lambda komposable.

Pada baris [13], `MaterialTheme` ( digunakan untuk menerapkan tema Material 3 pada UI, termasuk skema warna, tipografi, dan isi konten. Pada baris [14], `colorScheme = LightColorScheme` menetapkan bahwa tema yang digunakan adalah skema warna terang. Pada baris [15], `typography = Typography` menetapkan penggunaan gaya tipografi yang didefinisikan dalam objek `Typography` (yang diasumsikan telah dideklarasikan di file lain). Pada baris [16], `content = content` meneruskan konten UI komposable ke dalam tema `MaterialTheme`. Pada baris [17], penutup dari blok `MaterialTheme` dan fungsi `MovieListTheme`.

## 7. Theme.kt

Pada baris [1], `package com.example.moviescrollablelist.ui.theme` digunakan untuk menentukan namespace atau nama paket dari file tema dalam aplikasi

ini. Pada baris [3], import androidx.compose.material3.MaterialTheme digunakan untuk mengimpor komponen MaterialTheme dari Material 3 untuk menerapkan tema pada UI. Pada baris [4], import androidx.compose.material3.darkColorScheme digunakan untuk mengimpor fungsi pembuat skema warna gelap. Pada baris [5], import androidx.compose.material3.lightColorScheme digunakan untuk mengimpor fungsi pembuat skema warna terang. Pada baris [6], import androidx.compose.runtime.Composable digunakan untuk mendeklarasikan fungsi sebagai fungsi komposable dalam Jetpack Compose. Pada baris [7], import androidx.compose.ui.graphics.Color digunakan untuk mengimpor class Color untuk mendefinisikan warna dalam bentuk hexadecimal.

Pada baris [9], private val LightColors = lightColorScheme( . . . ) mendefinisikan variabel LightColors sebagai skema warna terang dengan warna primer dan sekunder khusus. Pada baris [10], primary = Color(0xFF6200EE) menetapkan warna ungu sebagai warna utama (primer) untuk tema terang. Pada baris [11], secondary = Color(0xFF03DAC6) menetapkan warna toska sebagai warna sekunder untuk tema terang. Pada baris [14], private val DarkColors = darkColorScheme( . . . ) mendefinisikan variabel DarkColors sebagai skema warna gelap dengan warna primer dan sekunder khusus. Pada baris [15], primary = Color(0xFFBB86FC) menetapkan warna ungu muda sebagai warna utama untuk tema gelap. Pada baris [16], secondary = Color(0xFF03DAC6) menetapkan warna toska sebagai warna sekunder untuk tema gelap. Pada baris [19], @Composable menandai fungsi MovieListTheme sebagai fungsi komposable yang dapat digunakan untuk membungkus UI dengan tema.

Pada baris [20], fun MovieListTheme( . . . ) mendefinisikan fungsi tema yang menerima parameter boolean darkTheme dan parameter content sebagai composable lambda. Pada baris [24], val colors = if (darkTheme) DarkColors else LightColors menentukan skema warna yang akan digunakan berdasarkan nilai darkTheme. Pada baris [26], MaterialTheme( digunakan untuk menerapkan tema

Material 3 pada komponen UI. Pada baris [27], `colorScheme = colors` menetapkan skema warna aktif sesuai pilihan terang atau gelap. Pada baris [28], `typography = Typography` menetapkan gaya tipografi yang digunakan dari objek `Typography`. Pada baris [29], `content = content` akan merender isi UI yang dibungkus oleh tema. Pada baris [31], menutup blok fungsi `MovieListTheme`.

## 8. Type.kt

Pada baris [1], `package com.example.moviescrollablelist.ui.theme` digunakan untuk menentukan namespace atau nama paket dari file tema aplikasi ini. Pada baris [3], `import androidx.compose.material3.Typography` digunakan untuk mengimpor class `Typography` dari Material 3 yang berfungsi untuk mengatur gaya teks aplikasi. Pada baris [4], `import androidx.compose.ui.text.TextStyle` digunakan untuk mengimpor class `TextStyle` yang mendeskripsikan gaya teks seperti ukuran, warna, dan ketebalan. Pada baris [5], `import androidx.compose.ui.text.font.FontFamily` digunakan untuk mengimpor class `FontFamily` yang digunakan untuk mengatur jenis huruf. Pada baris [6], `import androidx.compose.ui.text.font.FontWeight` digunakan untuk mengimpor class `FontWeight` yang digunakan untuk menentukan ketebalan huruf.

Pada baris [7], `import androidx.compose.ui.unit.sp` digunakan untuk mengimpor satuan ukuran sp (scale-independent pixels) yang biasa digunakan untuk ukuran teks. Pada baris [9], `val Typography = Typography( ... )` membuat instance dari `Typography` yang mendefinisikan gaya teks default yang akan digunakan di seluruh aplikasi. Pada baris [10], `bodyLarge = TextStyle( ... )` mendefinisikan gaya teks untuk `bodyLarge` (teks isi berukuran besar) dengan atribut tertentu. Pada baris [11], `fontFamily = FontFamily.Default` menetapkan font bawaan sistem sebagai jenis huruf. Pada baris [12], `fontWeight = FontWeight.Normal` menetapkan ketebalan teks menjadi normal. Pada baris [13], `fontSize = 16.sp` menetapkan ukuran huruf sebesar 16 scale-independent pixels. Pada baris [14], `lineHeight = 24.sp` menetapkan tinggi baris teks sebesar 24 sp.

Pada baris [15], `letterSpacing = 0.5.sp` menetapkan jarak antar huruf sebesar 0.5 sp. Pada baris [16], menutup definisi objek `TextStyle` dan sekaligus blok konfigurasi `Typography`.

## 9. MovieDetail.kt

Pada baris [1], `package com.example.moviescrollablelist.ui` digunakan untuk menentukan namespace atau nama paket dari file UI dalam aplikasi ini. Pada baris [3–23], berbagai fungsi dan class dari Jetpack Compose dan AndroidX diimpor untuk membangun tampilan antarmuka, seperti layout, ikon, gambar, teks, dan navigasi. Pada baris [25], anotasi `@Composable` menandakan bahwa fungsi berikut adalah komponen UI yang dapat dipanggil dalam hierarki tampilan Compose. Pada baris [26], `fun MovieDetail(navController: NavController, desc: String, image: Int)` mendefinisikan fungsi composable bernama `MovieDetail` dengan parameter navigasi, deskripsi, dan gambar. Pada baris [28], `Column` digunakan sebagai layout vertikal untuk menempatkan elemen UI secara berurutan dari atas ke bawah.

Pada baris [29–32], `modifier` diterapkan agar `Column` memenuhi ukuran layar, dapat scroll secara vertikal, dan memiliki padding 25dp. Pada baris [34], `IconButton` dibuat dengan aksi `onClick` untuk kembali ke layar sebelumnya menggunakan `navController.popBackStack()`. Pada baris [36], `Icon` ditampilkan dengan ikon panah kembali (ArrowBack) dari `Icons.Filled`. Pada baris [40], `ImageGlide` dipanggil untuk menampilkan gambar film dengan lebar penuh dan tinggi 300dp. Pada baris [41], `painterResource` mengambil gambar dari resource drawable berdasarkan ID. Pada baris [48], `Spacer` digunakan untuk memberi jarak vertikal sebesar 16dp setelah gambar.

Pada baris [49], `Text` digunakan untuk menampilkan deskripsi film dengan ukuran huruf 18sp. Pada baris [53], anotasi `@Composable` menandakan bahwa fungsi berikut juga merupakan komponen UI Compose. Pada baris [54–55], `fun ImageGlide(painter: Painter, contentDescription: Nothing?, modifier: Modifier)` mendefinisikan fungsi untuk menampilkan gambar. Pada

baris [56-58], `Image` digunakan untuk menampilkan gambar dengan parameter `painter`, `contentDescription`, dan `modifier` yang diberikan.

## 10. MovieList.kt

Pada baris [1], `package com.example.moviescrollablelist.ui` digunakan untuk menentukan namespace atau nama paket dari file UI aplikasi. Pada baris [3–4], `android.content.Intent` dan `android.net.Uri` diimpor untuk menangani intent ke browser (misalnya membuka link Letterboxd). Pada baris [5–15], berbagai komponen Compose UI seperti layout dan tampilan visual diimpor untuk menyusun elemen-elemen tampilan film. Pada baris [15–18], komponen `Material3` seperti `Button`, `Card`, dan `Text` diimpor untuk membuat elemen UI bergaya Material Design. Pada baris [19–21], `Composable` dan `Modifier` diimpor untuk menyusun tampilan secara deklaratif di Jetpack Compose. Pada baris [22], `LocalContext` digunakan untuk mengakses konteks Android saat ini di dalam `composable`. Pada baris [23–24], fungsi untuk mengakses resource gambar dan mengatur teks seperti font dan ukuran diimpor.

Pada baris [28], `NavController` digunakan untuk mengatur navigasi antar layar Compose. Pada baris [29], daftar data film `movieList` diimpor dari package data. Pada baris [31], anotasi `@Composable` menandai fungsi `MovieList` sebagai `composable function`. Pada baris [33], `val context = LocalContext.current` mengambil konteks aplikasi saat ini untuk digunakan dalam intent. Pada baris [35], `LazyColumn` digunakan untuk menampilkan daftar film secara efisien dengan scroll vertikal. Pada baris [36], `items(movieList.size)` menentukan jumlah item berdasarkan jumlah data di `movieList`. Pada baris [37], `val movie = movieList[index]` mengambil objek film berdasarkan indeks. Pada baris [38-44], `Card` digunakan untuk membungkus setiap item film dengan tampilan berbentuk kartu, sudut membulat, dan bayangan. Pada baris [45-49], `Row` digunakan untuk menyusun gambar dan teks secara horizontal, dengan padding 16dp.

Pada baris [50-57], `Image` menampilkan poster film dengan ukuran tetap lebar 100dp dan tinggi 140dp. Pada baris [58], `Spacer` memberi jarak horizontal antara gambar dan

kolom teks. Pada baris [59], Column mengatur teks dan tombol secara vertikal di samping gambar. Pada baris [60-61], Text menampilkan judul film dengan ukuran font 20sp dan gaya bold. Pada baris [62-67], Text kedua menampilkan deskripsi film dengan maksimum 3 baris dan elipsis jika terlalu panjang. Pada baris [68], Spacer menambahkan jarak vertikal 8dp sebelum tombol. Pada baris [70], Row digunakan untuk menyusun dua tombol secara horizontal. Pada baris [71-75], Button pertama membuka link Letterboxd melalui intent ke browser menggunakan ACTION\_VIEW dan Uri.parse(movie.url). Pada baris [82-83], Spacer menambahkan jarak horizontal 8dp antara tombol. Pada baris [84-85 dan 89 ], Button kedua melakukan navigasi ke halaman detail film dengan mengirim data deskripsi dan gambar melalui URI. Pada baris [87-88], Uri.encode(movie.description) memastikan deskripsi film dikodekan dengan benar sebelum dikirim lewat URI untuk menghindari error saat navigasi.

## **11. build.gradle.kts**

Pada baris [1-6], plugins digunakan untuk menambahkan plugin Android, Kotlin, Compose, dan kapt yang dibutuhkan untuk membangun aplikasi dengan Jetpack Compose dan Glide. Pada baris [8], android digunakan untuk mengatur konfigurasi dasar aplikasi seperti namespace, compileSdk, dan defaultConfig. Pada baris [9], namespace = "com.example.moviescrollablelist\_viewmodel" digunakan untuk menentukan nama paket unik aplikasi. Pada baris [10], compileSdk = 35 menetapkan versi SDK yang digunakan untuk meng-compile aplikasi. Pada baris [13], applicationId = "com.example.moviescrollablelist" menetapkan ID unik aplikasi untuk distribusi. Pada baris [14], minSdk = 30 menentukan versi Android minimum yang didukung aplikasi.

Pada baris [15], targetSdk = 35 menetapkan target versi Android untuk optimisasi kompatibilitas. Pada baris [16], versionCode = 1 menunjukkan versi internal aplikasi. Pada baris [17], versionName = "1.0" menunjukkan versi rilis aplikasi yang ditampilkan ke pengguna. Pada baris [19-20], testInstrumentationRunner digunakan untuk menentukan runner pengujian Android. Pada baris [22-23],

`buildTypes.release` mengatur konfigurasi untuk versi rilis aplikasi. Pada baris [25], `isMinifyEnabled = false` menunjukkan bahwa proguard belum diaktifkan untuk menyusutkan kode. Pada baris [26-30], `proguardFiles` menetapkan file konfigurasi ProGuard yang digunakan saat rilis.

Pada baris [34-37], `compileOptions` menetapkan kompatibilitas Java ke versi 11. Pada baris [39-41], `kotlinOptions.jvmTarget = "11"` menetapkan target JVM untuk Kotlin ke Java 11. Pada baris [43-45], `buildFeatures.compose = true` mengaktifkan Jetpack Compose sebagai fitur UI. Pada baris [47-49], `composeOptions.kotlinCompilerExtensionVersion` menentukan versi ekstensi compiler untuk Compose. Pada baris [52-80], `dependencies` digunakan untuk menyertakan library eksternal yang mendukung fungsionalitas utama aplikasi. Platform compose-bom digunakan agar versi library Compose tetap konsisten. Glide dan Coil digunakan untuk memuat gambar, Jetpack Compose dan Material3 untuk membangun antarmuka UI modern, Navigation Compose untuk navigasi antar layar, serta lifecycle-runtime untuk mengelola siklus hidup komponen. Bagian pengujian mencakup junit untuk unit test, dan Espresso serta ui-test-junit4 untuk instrumented test dan pengujian UI berbasis Compose. `debugImplementation` seperti ui-tooling dan ui-test-manifest membantu dalam proses pengembangan dan debugging UI.

## 12. MovieRepository.kt

Pada baris [1], package `com.example.moviescrollablelist_viewmodel.data` digunakan untuk menentukan lokasi paket file repository di dalam struktur proyek. Pada baris [3-4], `MovieData` dan `movieList` diimpor dari file yang sama untuk digunakan dalam class ini. Pada baris [6], `class MovieRepository` digunakan sebagai lapisan akses data untuk mengambil daftar film dan detail film berdasarkan ID. Pada baris [7], fungsi `getMovies()` mengembalikan seluruh daftar film dari `movieList`. Pada baris [12-13], fungsi `getMovieById(movieId: Int)` digunakan untuk mencari satu data film berdasarkan id, dan akan mengembalikan null jika tidak ditemukan.

## 13. MovieViewModel.kt

Pada baris [1], package com.example.moviescrollablelist\_viewmodel.data menetapkan lokasi file MovieViewModel dalam struktur paket aplikasi. Pada baris [3-4], ViewModel dan viewModelScope diimpor untuk membuat ViewModel yang dapat menjalankan coroutine secara aman terhadap siklus hidup. Pada baris [13-14], MovieRepository dan MovieData diimpor sebagai sumber data dan model yang digunakan. Pada baris [16-23], MutableStateFlow dan StateFlow dari Kotlin Coroutines digunakan untuk mengelola dan mengamati data secara reaktif. Pada baris [26], Log digunakan untuk mencatat log debugging. Pada baris [31], viewModelScope.launch digunakan untuk mengeksekusi coroutine dalam lingkup ViewModel. Pada baris [32], \_movieList menyimpan daftar film secara internal menggunakan MutableStateFlow. Pada baris [33-34], movieList mengekspos \_movieList sebagai StateFlow agar hanya bisa dibaca dari luar.

Pada baris [41-42], \_selectedMovie dan selectedMovie digunakan untuk menyimpan dan mengekspos data film yang dipilih.

#### 14. MovieViewModelFactory.kt

Pada baris [1], package com.example.moviescrollablelist\_viewmodel.data menyatakan lokasi file MovieViewModelFactory dalam struktur paket aplikasi. Pada baris [3-4], ViewModel dan ViewModelProvider diimpor untuk membuat dan menyediakan instance ViewModel. Pada baris [5], MovieRepository diimpor sebagai sumber data. Pada baris [8], anotasi @Suppress ("UNCHECKED\_CAST") digunakan untuk menonaktifkan peringatan cast tidak aman pada baris casting ViewModel. Pada baris [9-10], kelas MovieViewModelFactory didefinisikan dengan parameter repository yang akan diberikan ke MovieViewModel. Pada baris [11], fungsi create dioVERRIDE dari ViewModelProvider.Factory untuk membuat instance ViewModel. Pada baris [11-18], dilakukan pengecekan apakah modelClass adalah MovieViewModel. Jika sesuai, maka instance MovieViewModel dibuat dengan repository, lalu dikembalikan sebagai tipe T, jika modelClass bukan

`MovieViewModel`, maka dilemparkan `IllegalArgumentException`. Pada baris [22], fungsi `viewModelFactory()` digunakan untuk menyederhanakan pembuatan instance `MovieViewModelFactory`. Pada baris [23], objek repository dibuat dari `MovieRepository`. Pada baris [24], instance `MovieViewModelFactory` dikembalikan dengan menyertakan repository.

## 15. Penjelasan Debugger

Debugger adalah tool penting di Android Studio yang digunakan untuk menganalisis jalannya program secara detail seperti menelusuri dan memperbaiki kesalahan dalam aplikasi Android. Dengan debugger, kita bisa memeriksa nilai variabel secara langsung, memahami alur eksekusi program, dan mengetahui di mana bug terjadi secara real-time saat aplikasi berjalan.

Cara menggunakan debugger:

1. Set Breakpoint

Saya meletakkan *breakpoint* di dalam fungsi `val movie = movieList[index]` di `MovieList.kt`.

2. Klik ikon Debug di Android Studio.

3. Aplikasi akan berhenti saat mencapai breakpoint.

Fitur Debugger yang Saya Gunakan:

1. Resume Program (F9)

Melanjutkan eksekusi aplikasi ke breakpoint berikutnya atau hingga aplikasi selesai berjalan (Gambar 4).

2. Step Over (F8)

Menjalankan baris kode saat ini tanpa masuk ke dalam fungsi yang dipanggil. Cocok digunakan jika kita hanya ingin melihat hasil fungsi tanpa perlu masuk ke detailnya (Gambar 5).

3. Step Into (F7)

Masuk ke dalam fungsi yang sedang dipanggil, berguna untuk menelusuri isi fungsi secara detail (Gambar 6).

#### 4. Step Out (Shift+F8)

Keluar dari fungsi saat ini dan kembali ke kode pemanggilnya. Cocok saat kita sudah selesai mengecek isi fungsi (Gambar 7).

### D. Tautan Git

<https://github.com/aliyarfaura/Pemrograman-Mobile/tree/master/MODUL-4>

## SOAL 2

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

### A. Jawaban

Application class dalam arsitektur aplikasi itu seperti "otak pusat" atau base camp dari aplikasi Android. Application class akan berjalan sebelum activity, service, atau komponen lain muncul, dan cuma dibuat sekali selama aplikasi masih hidup. Fungsi dari Application class adalah untuk menyiapkan hal-hal global yang dibutuhkan oleh seluruh aplikasi. Misalnya: inisialisasi library, setup dependency injection, logging, atau untuk inisialisasi SDK sebelum UI Compose-nya muncul. Application class juga berfungsi sebagai tempat yang nyaman untuk menyimpan state global atau konfigurasi yang tidak berubah-ubah. Lalu, override onCreate()-nya untuk melakukan sesuatu saat aplikasi pertama kali dijalankan.

Contohnya:

```
class MyApp : Application() {  
    override fun onCreate() {  
        super.onCreate()  
  
        Log.d("MyApp", "Aplikasi dimulai")  
    }  
}
```

Jadi, Application class ini merupakan titik awal yang bisa dimanfaatkan sebagai setup awal dan untuk membuat hal penting yang siap dipakai semua komponen.