LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

Build a Scrollable List

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 3

Laporan Praktikum Pemrograman Mobile Modul 3: Build a Scrollable List ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum 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

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
    1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas 3.
    3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah 4. Terdapat 2 button dalam list, dengan fungsi berikut:
        a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
        b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item
    4. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
    5. 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
    6. Aplikasi menggunakan arsitektur single activity (satu activity memiliki beberapa fragment)
    7. Aplikasi berbasis XML harus menggunakan ViewBinding

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1 MainActivity.kt

```
1  package com.example.moviescrollablelist
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.Surface
8  import androidx.compose.ui.Modifier
```

```
import androidx.core.view.WindowCompat
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument
import com.example.moviescrollablelist.ui.MovieDetail
import com.example.moviescrollablelist.ui.MovieList // <-- Import
the MovieList composable
import com.example.moviescrollablelist.ui.theme.MovieListTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        WindowCompat.setDecorFitsSystemWindows(window, false)

        setContent {
            MovieListTheme {
                Surface(modifier = Modifier.fillMaxSize()) {
                    val navController = rememberNavController()
                    NavHost(navController = navController,
startDestination = "movieList") {
                        composable("movieList") {
                            MovieList(navController)
                        }
                        composable(
                            "movieDetail/{description}/{image}",
                            arguments = listOf(
                                navArgument("description") { type =
NavType.StringType },
                                navArgument("image") { type =
NavType.IntType }
                            )
                        ) { backStackEntry ->
```

```
                            val                desc                =
backStackEntry.arguments?.getString("description") ?: ""
                            val                image               =
backStackEntry.arguments?.getInt("image") ?: 0
                            MovieDetail(navController              =
navController, desc = desc, image = image)
                        }

                    }
                }
            }
        }
    }
}
```

2. MovieData.kt

Tabel 2. Source Code Jawaban Soal 1 MovieData.kt

```
package com.example.moviescrollablelist.data

data class MovieData(
    val name: String,
    val image: Int,
    val url: String,
    val description: String,
)
```

3. movieList.kt

Tabel 3. Source Code Jawaban Soal 1 movieList.kt

```
package com.example.moviescrollablelist.data

import com.example.moviescrollablelist.R

val movieList = listOf(
```

```
6       MovieData(
7           name = "Ada Apa Dengan Cinta?",
8           image = R.drawable.aadc,
9           url    =    "https://letterboxd.com/film/whats-up-with-
10  cinta/",
11          description = "Sinopsis: Ada Apa dengan Cinta? adalah
12  film drama remaja Indonesia yang mengisahkan Cinta, siswi SMA
13  populer yang hobi menulis puisi. Hidupnya berubah saat ia jatuh
14  cinta pada Rangga, siswa pendiam dari latar belakang berbeda.
15  Hubungan mereka yang tumbuh diam-diam membuat Cinta berkonflik
16  dengan   sahabat-sahabatnya,   memaksanya   memilih   antara
17  persahabatan atau cinta. Film ini menjadi ikon kisah cinta
18  remaja Indonesia dengan sentuhan puisi dan emosi yang mendalam.
19  Dibalut dengan alunan musik yang ikonik dan puisi-puisi yang
20  menyentuh hati. Ada Apa dengan Cinta? menjadi simbol romantika
21  masa muda yang membekas hingga kini."
22      ),
23      MovieData(
24          name = "Gie",
25          image = R.drawable.gie,
26          url = "https://letterboxd.com/film/gie/",
27          description = "Sinopsis: GIE adalah film biografi
28  tentang Soe Hok Gie, aktivis dan penulis idealis yang lantang
29  menyuarakan kebenaran di tengah gejolak politik era Soekarno
30  dan Soeharto. Sebagai mahasiswa yang kritis, Gie menolak segala
31  bentuk penindasan dan korupsi, meski harus berjuang seorang
32  diri. Film ini menggambarkan perjuangan moral, pencarian makna
33  hidup, dan semangat muda yang tak gentar melawan ketidakadilan.
34  Dengan latar sejarah Indonesia yang penuh gejolak, GIE menjadi
35  potret reflektif tentang suara kritis generasi muda, pentingnya
36  integritas moral, dan keberanian untuk berdiri sendiri dalam
37  memperjuangkan kebenaran, bahkan ketika itu berarti melawan
38  arus."
39
```

```
    ),
    MovieData(
        name = "Janji Joni",
        image = R.drawable.janji_joni,
        url = "https://letterboxd.com/film/jonis-promise/",
        description = "Sinopsis: Janji Joni mengisahkan Joni,
seorang pengantar gulungan film yang berjuang menepati janji
kepada gadis impiannya untuk mengantar film tepat waktu. Namun,
di tengah kota yang penuh kekacauan dan kejadian tak terduga,
misi sederhana itu berubah menjadi petualangan kocak dan penuh
tantangan. Dengan gaya penceritaan yang segar, musik yang
catchy, dan nuansa urban yang kuat, Janji Joni menyuguhkan kisah
tentang ketekunan, integritas, dan usaha seorang pria biasa yang
ingin menepati janji demi cinta yang tumbuh dalam sekejap."
    ),
    MovieData(
        name = "Marmut Merah Jambu",
        image = R.drawable.marmut_merah_jambu,
        url     =     "https://letterboxd.com/film/pink-guinea-
pig/details/",
        description    =    "Sinopsis:    Marmut    Merah    Jambu
menceritakan Dika, siswa SMA canggung yang diam-diam menyukai
gadis bernama Ina. Untuk menarik perhatiannya, ia membentuk Trio
Detektif bersama dua sahabatnya dan mulai menyelidiki kasus
misterius \"Demonic Pink Guinea Pig\" yang menghebohkan
sekolah. Di tengah penyelidikan yang penuh kekonyolan dan
kejadian absurd, Dika justru menemukan makna persahabatan,
penerimaan diri, dan cinta remaja yang tak selalu berjalan
sesuai harapan. Film ini menyuguhkan kisah cinta pertama dengan
humor khas Raditya Dika yang segar dan menghibur. "
    ),
    MovieData(
        name = "Aruna dan Lidahnya",
        image = R.drawable.aruna,
```

```
        url = "https://letterboxd.com/film/aruna-her-palate/",
        description = "Sinopsis: Aruna dan Lidahnya mengisahkan
Aruna, seorang epidemiolog yang gemar makan, saat ia melakukan
perjalanan ke berbagai daerah untuk menyelidiki kasus flu
burung. Bersama dua sahabatnya—seorang koki dan kritikus
makanan—perjalanan dinas itu berubah menjadi petualangan
kuliner yang penuh rasa, tawa, dan konflik. Di tengah sajian
makanan khas Nusantara, Aruna mulai menyadari makna cinta,
persahabatan, dan pencarian jati diri. Lewat visual yang
menggoda selera dan dialog yang cerdas, Aruna dan Lidahnya
menyuguhkan lebih dari sekadar kisah cinta dan makanan—ini
adalah perjalanan emosional yang mempertemukan rasa, memori,
dan hati. "
    ),
    MovieData(
        name = "Ali dan Ratu Ratu Queens",
        image = R.drawable.queens,
        url   =   "https://letterboxd.com/film/ali-ratu-ratu-
queens/",
        description = "Sinopsis: Ali dan Ratu-Ratu Queens
mengikuti perjalanan seorang remaja bernama Ali yang, setelah
kehilangan ayahnya, pergi ke New York untuk mencari ibu
kandungnya yang telah lama terpisah darinya. Di kota besar itu,
Ali bertemu dengan sekelompok wanita Indonesia yang memberinya
dukungan dan rasa kebersamaan. Di tengah pencariannya, Ali
menemukan cinta pertama dan membangun hubungan yang mengubah
hidupnya. Sebuah kisah tentang keluarga, cinta, dan pencarian
jati diri yang penuh kehangatan. "
    ),
    MovieData(
        name = "Mencuri Raden Saleh",
        image = R.drawable.raden_saleh,
        url   =   "https://letterboxd.com/film/stealing-raden-
saleh/",
```

```
        description = "Sinopsis: Mencuri Raden Saleh
mengisahkan seorang ahli pembuat lukisan palsu yang
merencanakan pencurian terbesar untuk menyelamatkan ayahnya. Ia
membentuk tim yang terdiri dari para ahli untuk mencuri lukisan
Raden Saleh yang sangat berharga. Dalam perjalanan yang penuh
ketegangan dan rintangan, mereka menghadapi tantangan dari
pihak berwenang dan musuh-musuh yang siap menggagalkan misi
mereka. Sebuah film aksi thriller yang menggali tema
persahabatan, pengorbanan, dan moralitas, dengan alur cerdas
dan penuh kejutan."
    ),
    MovieData(
        name = "The Big 4",
        image = R.drawable.big_4,
        url = "https://letterboxd.com/film/the-big-4/",
        description = "Sinopsis: The Big 4 mengisahkan Dina,
seorang detektif perempuan yang teguh mengikuti aturan, namun
harus bergabung dengan empat pembunuh bayaran yang eksentrik
dan sedang terpuruk dalam hidup mereka. Ketika ayah Nina dibunuh
dengan cara misterius, ia terpaksa bekerja sama dengan tim yang
kacau ini untuk mengungkap kebenaran. Dalam penyelidikan yang
penuh kekacauan, aksi, dan komedi, mereka mulai menggali
konspirasi besar yang melibatkan dunia kriminal yang berbahaya.
Meskipun sering bertentangan, Dina dan timnya belajar untuk
saling melengkapi dan bekerja sama, membentuk ikatan yang tak
terduga dalam upaya mereka mencari keadilan."
    ),
    MovieData(
        name = "A+",
        image = R.drawable.aplus,
        url = "https://letterboxd.com/film/a-2023/",
        description = "Sinopsis: Berjuang dari barisan try out
untuk menembus peringkat pertama, Kaliypso Dirgantari harus
menghadapi empat besar pemegang tahta di SMA Bina Indonesia: Re
```

```
Dirgantara, Kenan Aditya, Adinda Aletheia dan Aurora Calista.
Masing-masing didorong oleh motivasi mereka sendiri, kelimanya
bersaing ketat untuk mendapatkan peringkat paralel, setidaknya
sampai rahasia epik terkait sistem sekolah terungkap."
    ),
    MovieData(
        name = "Gadis Kretek",
        image = R.drawable.gadis_kretek,
        url   =   "https://letterboxd.com/film/cigarette-girl-
2023/",
        description = "Sinopsis: Soeraja merupakan pemilik
pabrik kretek Djagad Raya yang sedang sekarat. Namun, ia justru
ingin bertemu perempuan yang bukan istrinya, yakni Jeng Yah.
Sang istri pun cemburu karena hal tersebut merupakan permintaan
terakhir suaminya. Akan tetapi, ketiga anak Soeraja, yakni
Lebas, Karim, dan Tegar tetap berusaha mencari keberadaan Jeng
Yah ke pelosok Pulau Jawa. Ketika berada dalam perjalanan untuk
mencari jejak Jeng Yah, mereka bertemu buruh batil yang menguak
asal-usul Kretek Djagad Raya hingga menjadi kretek nomor satu
di Indonesia dan juga mengetahui kisah cinta sang ayah dengan
Jeng Yah."
    ),
    MovieData(
        name = "Azzamine",
        image = R.drawable.azzamine,
        url = "https://letterboxd.com/film/azzamine/",
        description = "Sinopsis: Jasmine, seorang mahasiswi
dengan orang tua Muslim konservatif, merasa hidupnya terbatas.
Ia ingin menikmati masa mudanya dengan Deka, pacarnya yang
sangat mencintainya. Namun, orang tua Jasmine tidak setuju
dengan hubungan mereka dan berusaha menjodohkannya dengan
Azzam, seorang pemuda yang tampak sempurna, berperilaku saleh
seperti “Ustad.”Untuk menolak jodohkan ini, Jasmine berusaha
menjauhkan Azzam, namun Azzam justru dengan tenang memberi
```

```
perhatian tanpa membuat keributan. Alih-alih menjauh, Azzam membimbingnya dengan lembut, dan seiring waktu, Jasmine merasa hatinya terbagi setengah untuk Deka yang sangat mencintainya, dan setengah untuk Azzam yang baru dikenalnya namun perlahan membuatnya jatuh hati."
    ),
    MovieData(
        name = "Home Sweet Loan",
        image = R.drawable.home_sweet_loan,
        url = "https://letterboxd.com/film/home-sweet-loan/",
        description = "Sinopsis: Home Sweet Loan mengisahkan perjuangan Kaluna, seorang wanita muda yang bekerja di bank dengan gaji pas-pasan. Kaluna bermimpi memiliki rumah sendiri di Jakarta, namun menghadapi berbagai tantangan finansial.  Ia tinggal bersama keluarga besarnya yang terdiri dari orang tua, dua kakak beserta keluarga mereka dalam satu rumah yang sempit. Bersama tiga sahabatnya Tanisha, Kamamiya, dan Danan - Kaluna berusaha keras mencari hunian terjangkau di pinggiran Jakarta. Ia harus memilih antara memperjuangkan mimpinya sendiri atau membantu keluarganya yang terlilit masalah utang. Film ini menggambarkan realita generasi sandwich dalam menghadapi tekanan finansial dan keluarga. Mereka rela mengerem pengeluaran dan mencari kerja sampingan demi mewujudkan impian tersebut. Namun, Kaluna dihadapkan pada dilema saat kondisi keuangan keluarganya memburuk dan membutuhkan bantuan.  "
    ),
    MovieData(
        name = "Jumbo",
        image = R.drawable.jumbo,
        url = "https://letterboxd.com/film/jumbo-2025/",
        description = "Sinopsis: Jumbo bercerita tentang Don, seorang anak yatim piatu berusia 10 tahun dengan tubuh besar yang sering diremehkan. Don menemukan pelarian dan inspirasi dalam buku dongeng warisan orang tuanya, yang penuh cerita
```

```
ajaib. Untuk membuktikan dirinya, Don bertekad mengikuti
pertunjukan bakat dengan sandiwara yang terinspirasi dari buku
tersebut. Namun, mimpi Don dihina oleh teman-temannya, dan buku
dongengnya dicuri oleh perundung. Untungnya, Don mendapat
dukungan dari Oma dan sahabat-sahabatnya. Dalam usahanya
mendapatkan kembali bukunya, Don bertemu dengan Meri, seorang
anak dari dunia lain yang butuh bantuan untuk menemukan orang
tuanya. Petualangan penuh keajaiban pun dimulai, mengajarkan
arti persahabatan, keberanian, dan kepercayaan diri. "
    ),
)
```

4. glideimage.kt

Tabel 4. Source Code Jawaban Soal 1 glideimage.kt

```
package com.example.moviescrollablelist.ui.components

import android.widget.ImageView
import androidx.annotation.DrawableRes
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.viewinterop.AndroidView
import com.bumptech.glide.Glide

@Composable
fun GlideImage(
    @DrawableRes resId: Int,
    contentDescription: String?,
    modifier: Modifier = Modifier
) {
    AndroidView(
        factory = { context ->
            ImageView(context).apply {
                scaleType = ImageView.ScaleType.FIT_CENTER

```

```
                contentDescription?.let                         {
this.contentDescription = it }
            }
        },
        update = { imageView ->
            Glide.with(imageView.context)
                .load(resId)
                .into(imageView)
        },
        modifier = modifier
        )
}
```

5. Color.kt

Tabel 5. Source Code Jawaban Soal 1 Color.kt

```
package com.example.moviescrollablelist.ui.theme

import androidx.compose.ui.graphics.Color

val Purple80 = Color(0xFFD0BCFF)
val PurpleGrey80 = Color(0xFFCCC2DC)
val Pink80 = Color(0xFFEFB8C8)

val Purple40 = Color(0xFF6650a4)
val PurpleGrey40 = Color(0xFF625b71)
val Pink40 = Color(0xFF7D5260)
```

6. MovieListTheme.kt

Tabel 6. Source Code Jawaban Soal 1 MovieListTheme.kt

```
package com.example.moviescrollablelist.ui.theme

import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
```

```
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable

private val DarkColorScheme = darkColorScheme()
private val LightColorScheme = lightColorScheme()

@Composable
fun MovieListTheme(content: @Composable () -> Unit) {
    MaterialTheme(
        colorScheme = LightColorScheme,
        typography = Typography,
        content = content
    )
}
```

7. Theme.kt

Tabel 7. Source Code Jawaban Soal 1 Theme.kt

```
package com.example.moviescrollablelist.ui.theme

import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.ui.graphics.Color

private val LightColors = lightColorScheme(
    primary = Color(0xFF6200EE),
    secondary = Color(0xFF03DAC6),
)

private val DarkColors = darkColorScheme(
    primary = Color(0xFFBB86FC),
    secondary = Color(0xFF03DAC6),
)
```

```
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.moviescrollablelist.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.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)
```

```
    ) {
        IconButton(onClick = { navController.popBackStack() })
{
            Icon(imageVector     =     Icons.Filled.ArrowBack,
contentDescription = "Back")
        }

        ImageGlide(
            painter = painterResource(id = image),
            contentDescription = null,
            modifier = Modifier
                .fillMaxWidth()
                .height(300.dp)
        )

        Spacer(modifier = Modifier.height(16.dp))
        Text(text = desc, fontSize = 18.sp)
    }
}

@Composable
fun ImageGlide(painter: Painter, contentDescription: Nothing?,
modifier: Modifier) {
    Image(painter    =    painter,    contentDescription    =
contentDescription, modifier = modifier)
}
```

10. MovieList.kt

Tabel 10. Source Code Jawaban Soal 1 MovieList.kt

```
package com.example.moviescrollablelist.ui

import android.content.Intent
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.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(
```

```
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(vertical = 8.dp),
                shape = RoundedCornerShape(16.dp),
                elevation = CardDefaults.cardElevation(4.dp)
            ) {
                Row(
                    verticalAlignment = Alignment.CenterVertically,
                    modifier = Modifier.padding(16.dp)
                ) {
                    Image(
                        painter = painterResource(id = movie.image),
                        contentDescription = null,
                        modifier = Modifier
                            .width(100.dp)
                            .height(140.dp)
                    )
                    Spacer(modifier = Modifier.width(16.dp))
                    Column(modifier = Modifier.weight(1f)) {
                        Text(text = movie.name, fontSize = 20.sp, fontWeight = FontWeight.Bold)
                        Text(
                            text = movie.description,
                            fontSize = 14.sp,
                            maxLines = 3,
                            overflow = TextOverflow.Ellipsis
                        )
                        Spacer(modifier = Modifier.height(8.dp))
                        Row {
                            Button(
                                onClick = {
```

```
                                    val         intent          =
Intent(Intent.ACTION_VIEW, Uri.parse(movie.url))

context.startActivity(intent)
                                    }
                                ) {
                                    Text("Letterboxd")
                                }
                                Spacer(modifier                     =
Modifier.width(8.dp))
                                Button(
                                    onClick = {

navController.navigate("movieDetail/${Uri.encode(movie.descript
ion)}/${movie.image}")
                                    }
                                ) {
                                    Text("Detail")
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

11. build.gradle.kts

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

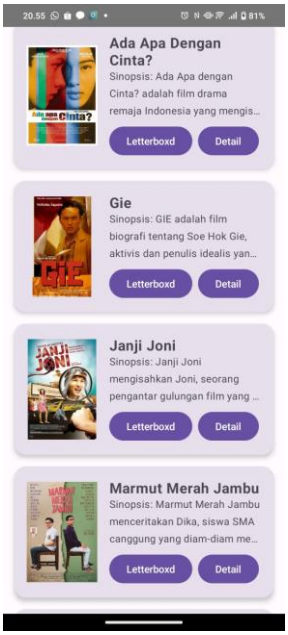
```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose)
```

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

```
    kotlinOptions {
        jvmTarget = "11"
    }

    buildFeatures {
        compose = true
    }

    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.3"
    }
}

dependencies {
    implementation(platform("androidx.compose:compose-
bom:2023.03.00"))

androidTestImplementation(platform("androidx.compose:compose-
bom:2023.03.00"))

    implementation("com.github.bumptech.glide:glide:4.15.1")
    kapt("com.github.bumptech.glide:compiler:4.15.1")

    implementation("androidx.compose.ui:ui")
    implementation("androidx.compose.material3:material3")
    implementation("androidx.navigation:navigation-
compose:2.7.0")
    implementation("io.coil-kt:coil-compose:2.4.0")
    implementation("androidx.activity:activity-compose:1.7.2")
    implementation("androidx.lifecycle:lifecycle-runtime-
ktx:2.6.1")
    implementation("androidx.compose.ui:ui-tooling-preview")
    debugImplementation("androidx.compose.ui:ui-tooling")

```

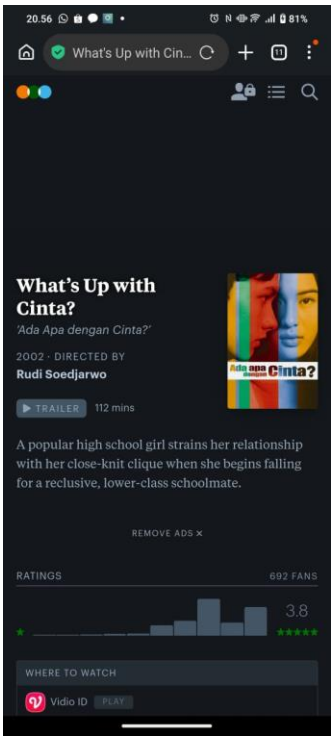
```
73      testImplementation("junit:junit:4.13.2")
74      androidTestImplementation("androidx.test.ext:junit:1.1.5")
75      androidTestImplementation("androidx.test.espresso:espresso-
76  core:3.5.1")
77      androidTestImplementation("androidx.compose.ui:ui-test-
78  junit4")
79      debugImplementation("androidx.compose.ui:ui-test-manifest")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

B. Output Program



Gambar 1. Screenshot Output Tampilan Halaman List Soal 1

Gambar 2. Screenshot Output Tampilan Halaman Detail Soal 1



Gambar 3. Screenshot Output Tampilan dari Link Letterboxd

C. Pembahasan

1. MainActivity.kt

Pada baris [1], `package com.example.moviescrollablelist` digunakan untuk menentukan namespace atau nama paket dari aplikasi. Pada baris [3], `import android.os.Bundle` digunakan untuk mengakses kelas Bundle yang berguna menyimpan state aplikasi. Pada baris [4], `import androidx.activity.ComponentActivity` mengimpor kelas dasar Activity yang mendukung Jetpack Compose. Pada baris [5], `import androidx.activity.compose.setContent` digunakan untuk menampilkan UI berbasis Compose di dalam activity. Pada baris [6], `import androidx.compose.foundation.layout.fillMaxSize` digunakan untuk mengisi seluruh ukuran tampilan dengan layout.

Pada baris [7], `import androidx.compose.material3.Surface` mengimpor komponen permukaan latar belakang dari Material 3. Pada baris [8], `import androidx.compose.ui.Modifier` digunakan untuk menerapkan modifikasi pada komponen UI Compose. Pada baris [9], `import androidx.core.view.WindowCompat` digunakan untuk mengatur kompatibilitas tampilan jendela, seperti inset sistem. Pada baris [10], `import androidx.navigation.NavType` digunakan untuk menentukan tipe argumen dalam navigasi.

Pada baris [11], `import androidx.navigation.compose.NavHost` digunakan untuk membuat container navigasi berbasis Compose. Pada baris [12], `import androidx.navigation.compose.composable` digunakan untuk mendefinisikan rute layar (composable) di NavHost. Pada baris [13], `import androidx.navigation.compose.rememberNavController` digunakan untuk menyimpan dan mengelola status navigasi. Pada baris [14], `import androidx.navigation.navArgument` digunakan untuk mendefinisikan argumen yang dikirim antar layar. Pada baris [15], `import com.example.moviescrollablelist.ui.MovieDetail` digunakan untuk mengakses tampilan detail film. Pada baris [16], `import com.example.moviescrollablelist.ui.MovieList` digunakan untuk

mengakses tampilan daftar film. Pada baris [17], `import com.example.moviescrollablelist.ui.theme.MovieListTheme` digunakan untuk menerapkan tema kustom aplikasi.

Pada baris [20], `class MainActivity : ComponentActivity()` mendefinisikan activity utama yang mewarisi dari ComponentActivity. Pada baris [21], `override fun onCreate` digunakan untuk menulis ulang fungsi lifecycle saat activity dibuat. Pada baris [22], `super.onCreate(savedInstanceState)` memanggil implementasi onCreate dari superclass. Pada baris [23], `WindowCompat.setDecorFitsSystemWindows(window, false)` digunakan untuk membuat konten bisa menggantikan area sistem (fullscreen). Pada baris [25], `setContent` digunakan untuk menampilkan komposisi UI berbasis Jetpack Compose. Pada baris [26], `MovieListTheme` digunakan untuk membungkus tampilan dengan tema aplikasi.

Pada baris [27], `Surface(modifier = Modifier.fillMaxSize())` digunakan untuk membuat latar belakang utama yang memenuhi layar. Pada baris [28], `val navController = rememberNavController()` menyimpan dan mengingat kontrol navigasi antar layar. Pada baris [29-30], `NavHost(navController = ..., startDestination = "movieList")` mendefinisikan host navigasi dengan rute awal "movieList". Pada baris [31], `composable("movieList")` digunakan untuk menampilkan layar daftar film. Pada baris [32], `MovieList(navController)` menampilkan composable MovieList dan mengirim navController untuk navigasi. Pada baris [34-36], `composable("movieDetail/{description}/{image}", ...)` mendefinisikan rute detail film dengan dua argumen dinamis.

Pada baris [36], `arguments = listOf(...)` menyatakan bahwa rute menerima argumen "description" (String) dan "image" (Int). Pada baris [43], `val desc = ...` mengambil argumen description dari backStackEntry atau default ke string kosong. Pada baris [45], `val image = ...` mengambil argumen image dari backStackEntry atau

default ke 0. Pada baris [47], `MovieDetail(...)` menampilkan tampilan detail dengan data yang telah diambil dari argumen.

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], `name = "Ada Apa Dengan Cinta?"` menetapkan judul film yang ditampilkan. Pada baris [8], `image = R.drawable.aadc` menetapkan referensi resource gambar drawable untuk film tersebut. Pada baris [9-10], `url = "https://letterboxd.com/film/whats-up-with-cinta/"` adalah

tautan eksternal ke halaman film. Pada baris [11], `description = "..."`, menyimpan deskripsi panjang film dalam bentuk string, menjelaskan sinopsis dan tema utama. Pada baris [22] 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 [220]), 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 resId: Int` menetapkan bahwa parameter resId 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 Compose. 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(0xFFEFB8C8)` 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 discroll 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], `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], `NavHostController` 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 [84-89], `Button` kedua melakukan navigasi ke halaman detail film dengan mengirim data deskripsi dan gambar melalui URI.

11. MovieList.kt

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"` 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.

D. Tautan Git

https://github.com/aliyarfnaura/Pemrograman-Mobile

SOAL 2

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

A. Jawaban

RecyclerView masih digunakan sampai sekarang meskipun kelihatannya memiliki kode yang panjang dan penuh boiler-plate karena sebenarnya dia sudah jadi andalan sejak lama di dunia Android development. Banyak aplikasi besar yang dibangun sebelum Jetpack Compose muncul, dan mereka sudah pakai RecyclerView dari awal. Jadi daripada harus migrasi total ke Compose (yang bisa makan waktu dan tenaga), mereka tetap bertahan dengan yang sudah stabil dan teruji.

Selain itu, RecyclerView punya fleksibilitas tinggi. Jika ingin membuat list biasa, grid, carousel, atau bahkan list yang bisa di-drag dan swipe, RecyclerView bisa semuanya. Setup-nya memang panjang, tapi setelah jalan, kontrolnya lebih detail. Ini berbeda dengan LazyColumn di Compose, yang memang lebih simpel dan elegan tapi belum sefleksibel RecyclerView untuk beberapa use case kompleks.

RecyclerView juga punya dukungan luas dari library-library pihak ketiga, dokumentasi, dan komunitas. Jadi kalau ada masalah, solusinya biasanya sudah ada di Stack Overflow atau GitHub. Plus, untuk aplikasi yang masih harus support Android versi lama, RecyclerView jadi solusi yang lebih aman dibanding langsung pakai Compose yang butuh API level tertentu.

Jadi walaupun Compose dan LazyColumn makin populer karena sintaksnya yang lebih modern dan clean, RecyclerView masih bertahan karena dia powerful, sudah matang, dan banyak dipakai di project-project di duni.