LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

Build a Scrollable List

Oleh:

Muhammad Adh-Dhiya'Us Salim NIM. 2310817210022

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 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 : Muhammad Adh-Dhiya'Us Salim
NIM : 2310817210022

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Ir. Eka Setya Wijaya, S.T., M.Kom. NIP. 198205082008011010

DAFTAR ISI

DAFTAR GAMBAR

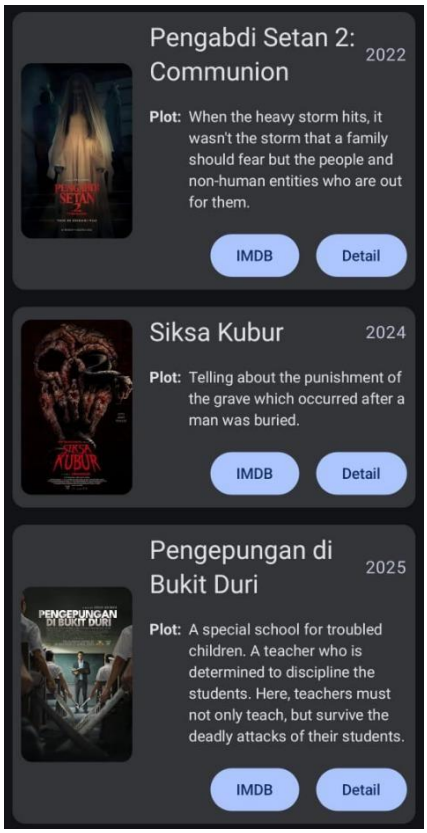
DAFTAR TABEL

SOAL 1

Soal Praktikum:

1. Buatlah sebuah aplikasi Android menggunakan XML dan Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
    1. List menggunakan fungsi RecyclerView (XML) dan LazyColumn (Compose)
    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
    3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah
    4. Terdapat 2 button dalam list, dengan fungsi berikut:
        a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
        b. Button kedua menggunakan Navigation component untuk membuka laman detail item
    5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
    6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
    7. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)
    8. Aplikasi berbasis XML harus menggunakan ViewBinding
2. Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

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



Gambar 1 Contoh UI List

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

Gambar 2 Contoh UI Detail

A. Source Code
Data/MovieRepository/ Compose

```
package com.example.scrollablelist_modul3.data


import com.example.scrollablelist_modul3.R
import com.example.scrollablelist_modul3.model.Movie

object MovieRepository {
    fun getMovies(): List<Movie> {
        return listOf(
            Movie(
                title = "Uncanny Counter Season 1",
                year = "2020",
                plot = "Para Counter yang karyawan toko mi di siang hari dan pemburu iblis di malam hari, memakai berbagai kemampuan khusus untuk memburu roh-roh jahat yang mengincar manusia.",
                imdb = "https://www.imdb.com/title/tt13273826/",
                posterResId = R.drawable.dukun
            ),
            Movie(
                title = "Sweet Home Season 1",
                year = "2020",
                plot = "Saat banyak manusia berubah menjadi monster ganas dan dunia terpuruk ke dalam teror, sekelompok penyintas berjuang untuk hidup—dan mempertahankan kemanusiaan mereka..",
                imdb =
"https://www.imdb.com/title/tt11612120/?ref_=nv_sr_srsg_0_tt_8_nm_0_in_0_q_Sweet%2520Home",
                posterResId = R.drawable.rm
            ),
            Movie(
                title = "Money Heist Korea",
                year = "2022",
                plot = "Pencuri menguasai gedung percetakan uang milik Korea bersatu dan menawan banyak sandera. Polisi harus menghentikan aksi mereka, serta dalang yang bersembunyi di baliknya.",
                imdb =
"https://www.imdb.com/title/tt13696452/?ref_=nv_sr_srsg_0_tt_8_nm_0_in_0_q_Money%2520Heist%2520Ko",
                posterResId = R.drawable.duitheist
            ),
            Movie(
                title = "My Name",
                year = "2021",
                plot = "Setelah ayahnya dibunuh, seorang wanita yang ingin membalas dendam memutuskan untuk memercayai bos kriminal yang berkuasa—dan memasuki kepolisian atas petunjuk pria itu.",
                imdb =
"https://www.imdb.com/title/tt12940504/?ref_=nv_sr_srsg_3_tt_8_nm_0_in_0_q_My%2520Name",
                posterResId = R.drawable.ngaran
            ),
            Movie(
                title = "The 8 Show",
```

```
                year                    =                    "2024",
                plot = "Delapan orang yang terjebak di gedung delapan
lantai misterius mengikuti acara yang menarik tetapi berbahaya. Dengan
berjalannya   waktu,   mereka   akan   mendapat   hadiah   uang.",
                imdb                                                 =
"https://www.imdb.com/title/tt30423279/?ref_=nv_sr_srsg_0_tt_8_nm_0_
in_0_q_The%25208%2520Show",
                posterResId          =          R.drawable.delapan
            ),
            Movie(
                title       =       "Kingdom       Season       1",
                year                    =                    "2020",
                plot = "Saat rumor aneh tentang raja yang sakit
menguasai kerajaan, putra mahkota menjadi satu-satunya harapan mereka
untuk   melawan   wabah   misterius   yang   menjangkiti   negeri.",
                imdb                                                 =
"https://www.imdb.com/title/tt6611916/?ref_=nv_sr_srsg_6_tt_8_nm_0_i
n_0_q_Kingdom",
                posterResId            =            R.drawable.raja
            ),
            Movie(
                title           =           "The           Frog",
                year                    =                    "2024",
                plot = "Di musim panas nan damai, seorang wanita
misterius masuk ke sebuah penginapan—memicu beragam peristiwa yang
mengganggu kehidupan si pemilik dan orang-orang di sekitarnya.",
                imdb   =   "https://www.imdb.com/title/tt26767508/",
                posterResId            =            R.drawable.frog
            ),
            Movie(
                title     =     "All     of     Us     Are     Dead",
                year                    =                    "2022",
                plot = "Sebuah SMA menjadi titik nol merebaknya wabah
virus zombi. Para murid yang terperangkap pun harus berjuang untuk
kabur   jika   tak   mau   terinfeksi   dan   berubah   menjadi   buas.",
                imdb   =   "https://www.imdb.com/title/tt14169960/",
                posterResId          =          R.drawable.allofus
            ),
            Movie(
                title          =          "Taxi          Driver",
                year                    =                     "2021",
                plot = "Seorang sopir taksi di Seoul mengantar seorang
wartawan Jerman untuk menyelidiki isu kerusuhan di Gwangju tanpa tahu
apa    yang    menanti    mereka.    Berdasarkan    kisah    nyata.",
                imdb   =   "https://www.imdb.com/title/tt13759970/",
                posterResId            =            R.drawable.taxi
            ),
            Movie(
                title       =       "A       Killer       Paradox",
                year                    =                     "2024",
                plot   =   "Saat   suatu   pembunuhan   tak   disengaja
memunculkan pembunuhan serupa lainnya, pemuda biasa ini terjebak dalam
aksi kucing-kucingan tanpa henti dengan detektif yang lihai.",
                imdb   =   "https://www.imdb.com/title/tt28642796/",
                posterResId          =          R.drawable.paradox
            )
```

```
        )
    }
}
```

Tabel 1 Soruce Code Soal 1 - Compose

Model / Movie / Compose

```
package com.example.scrollablelist_modul3.model

data class Movie(
    val title: String,
    val year: String,
    val plot: String,
    val imdb: String,
    val posterResId: Int
)
```

Tabel 2 Source Code Soal 1 - Compose

navigation / AppNavigation / Compose

```
package com.example.scrollablelist_modul3.navigation


import androidx.compose.runtime.Composable
import androidx.navigation.NavType
import androidx.navigation.navArgument
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.example.scrollablelist_modul3.ui.DetailScreen
import com.example.scrollablelist_modul3.ui.MovieListScreen
import com.example.scrollablelist_modul3.R


@Composable
fun AppNavigation() {
    val navController = rememberNavController()

    NavHost(navController = navController, startDestination =
Screen.MovieList.route) {
        composable(Screen.MovieList.route) {
            MovieListScreen(navController)
        }

        composable(
            route = Screen.Detail.route,
            arguments = listOf(
                navArgument("title") { type = NavType.StringType
},
                navArgument("year") { type = NavType.StringType
},
                navArgument("plot") { type = NavType.StringType
},
                navArgument("posterResId") { type =
NavType.IntType }
            )
        ) { backStackEntry ->
```

```
            val                         title                          =
backStackEntry.arguments?.getString("title")        ?:        ""
            val                         year                           =
backStackEntry.arguments?.getString("year")         ?:        ""
            val                         plot                           =
backStackEntry.arguments?.getString("plot")         ?:        ""
            val                   posterResId                          =
backStackEntry.arguments?.getInt("posterResId")                        ?:
R.drawable.ngaran

            DetailScreen(
                title                  =                  title,
                year                   =                   year,
                plot                   =                   plot,
                posterResId            =            posterResId)
        }
    }
}
```

Tabel 3 Source Code Soal 1 - Compose

navigation / Screen / Compose

```
1  package                  com.example.scrollablelist_modul3.navigation
2  import                                                 android.net.Uri
3
4  sealed       class       Screen(val       route:       String)       {
5      object          MovieList          :          Screen("movie_list")
6      object                            Detail                          :
7  Screen("movie_detail/{title}/{year}/{plot}/{posterResId}")             {
8          fun  createRoute(title:  String,  year:  String,  plot:  String,
9  posterResId:               Int):               String                {
               return
   "movie_detail/${Uri.encode(title)}/${Uri.encode(year)}/${Uri.encode(p
   lot)}/$posterResId"
           }
       }
   }
```

Tabel 4 Source Code Soal 1 - Compose

ui.theme / DetailScreen / Compose

```
1  package                    com.example.scrollablelist_modul3.ui
2
3  import                        androidx.compose.foundation.Image
4  import                     androidx.compose.foundation.layout.*
5  import     androidx.compose.foundation.shape.RoundedCornerShape
6  import                              androidx.compose.material3.*
7  import                      androidx.compose.runtime.Composable
8  import                              androidx.compose.ui.Modifier
9  import                            androidx.compose.ui.draw.clip
   import                   androidx.compose.ui.res.painterResource
   import                    androidx.compose.ui.res.stringResource
   import                              androidx.compose.ui.unit.dp
   import                     com.example.scrollablelist_modul3.R
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DetailScreen(title: String, year: String, plot: String,
posterResId: Int) {
    Scaffold(
        topBar = {
            TopAppBar(
                title = {
                    Text(text = "Movie Detail")
                }
            )
        }
    ) { paddingValues ->
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(paddingValues)
                .padding(16.dp)
        ) {
            Image(
                painter = painterResource(id = posterResId),
                contentDescription = null,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(550.dp)
                    .clip(RoundedCornerShape(16.dp))
            )
            Spacer(modifier = Modifier.height(16.dp))
            Text(
                text =
String.format(stringResource(R.string.movie_detail_title),
title, year),
                style = MaterialTheme.typography.headlineSmall
            )
            Spacer(modifier = Modifier.height(8.dp))
            Text(text =
"${stringResource(R.string.plot_label)}:\n$plot")
        }
    }
}
```

Tabel 5 Source Code Soal 1 - Compose

Ui.theme / MovieListScreen / Compose

```
1  package com.example.scrollablelist_modul3.ui
2
3  import androidx.compose.foundation.layout.*
4  import androidx.compose.foundation.lazy.LazyColumn
5  import androidx.compose.foundation.lazy.items
6  import androidx.compose.material3.*
7  import androidx.compose.runtime.Composable
8  import androidx.compose.ui.Modifier
9  import androidx.compose.ui.unit.dp
   import androidx.navigation.NavController
   import com.example.scrollablelist_modul3.data.MovieRepository
   import com.example.scrollablelist_modul3.navigation.Screen
   import androidx.compose.foundation.Image
```

```
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.ui.draw.clip
import androidx.compose.ui.res.painterResource
import android.content.Intent
import android.net.Uri
import androidx.compose.ui.platform.LocalContext


@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MovieListScreen(navController: NavController) {
    val movieList = MovieRepository.getMovies()
    val context = LocalContext.current

    Scaffold(
        topBar = {
            TopAppBar(
                title = {
                    Text(text = "KDrama Cinema")
                }
            )
        }
    ) { paddingValues ->
        LazyColumn(
            modifier = Modifier
                .fillMaxSize()
                .padding(paddingValues)
                .padding(8.dp)
        ) {
            items(movieList) { movie ->
                Card(
                    modifier = Modifier
                        .fillMaxWidth()
                        .padding(vertical = 8.dp),
                    shape = RoundedCornerShape(16.dp),
                    elevation =
CardDefaults.cardElevation(4.dp)
                ) {
                    Row(modifier = Modifier.padding(16.dp)) {
                        // Poster Film
                        Image(
                            painter = painterResource(id =
movie.posterResId),
                            contentDescription = null,
                            modifier = Modifier
                                .size(width = 100.dp, height =
150.dp)
                                .clip(RoundedCornerShape(8.dp))
                        )

                        Spacer(modifier =
Modifier.width(16.dp))

                        // Detail Film
                        Column(modifier = Modifier.weight(1f))
{
```

```
                            Row(
                                modifier                        =
Modifier.fillMaxWidth(),
                                horizontalArrangement           =
Arrangement.SpaceBetween
                            )                                   {
                                Text(
                                    text     =     movie.title,
                                    style                       =
MaterialTheme.typography.titleMedium
                                )
                                Text(
                                    text     =     movie.year,
                                    style                       =
MaterialTheme.typography.bodySmall
                                )
                            }

                            Spacer(modifier                     =
Modifier.height(8.dp))

                            Text(
                                text       =       movie.plot,
                                style                           =
MaterialTheme.typography.bodySmall,
                                maxLines          =          3
                            )

                            Spacer(modifier                     =
Modifier.height(8.dp))

                            Row(
                                modifier                        =
Modifier.fillMaxWidth(),
                                horizontalArrangement           =
Arrangement.End
                            )                                   {
                                Button(
                                    onClick         =         {
                                        val       intent       =
Intent(Intent.ACTION_VIEW,              Uri.parse(movie.imdb))

context.startActivity(intent)
                                    },
                                    modifier                    =
Modifier.padding(end                      =                   8.dp)
                                )                               {
                                    Text(text     =     "IMDB")
                                }

                                Button(
                                    onClick         =         {
                                        navController.navigate(

Screen.Detail.createRoute(
                                                movie.title,
```

```
                                                    movie.year,
                                                    movie.plot,

movie.posterResId
                                                )
                                            )
                                        }
                                    )                                   {
                                        Text(text    =    "Detail")
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

Tabel 6 Source Code Soal 1 - Compose

MainActivity / Compose

```
1  package                              com.example.scrollablelist_modul3
2
3  import                                              android.os.Bundle
4  import                              androidx.activity.ComponentActivity
5  import                             androidx.activity.compose.setContent
6  import    com.example.scrollablelist_modul3.navigation.AppNavigation
7  import
8  com.example.scrollablelist_modul3.ui.theme.ScrollableListModul3Theme
9
   class        MainActivity        :        ComponentActivity()         {
       override    fun    onCreate(savedInstanceState:    Bundle?)     {
           super.onCreate(savedInstanceState)
           setContent                                                    {
               ScrollableListModul3Theme                                 {
                   AppNavigation()
               }
           }
       }
   }
```

Tabel 7 Source Code Soal 1 - Compose

Data / DataSource / XML

```
package com.example.m3xml.data

import com.example.m3xml.R

object DataSource {
    fun getFilms(): List<Film> {
        return listOf(
            Film(
                1,
                "Unchanny Counter Season 1",
                "2020",
                "Para Counter yang karyawan toko mi di siang hari dan pemburu iblis di malam hari, memakai berbagai kemampuan khusus untuk memburu roh-roh jahat yang mengincar manusia.",
                R.drawable.dukun,
                "https://www.imdb.com/title/tt13273826/"
            ),
            Film(
                2,
                "Sweet Home",
                "2020",
                "Saat banyak manusia berubah menjadi monster ganas dan dunia terpuruk ke dalam teror, sekelompok penyintas berjuang untuk hidup—dan mempertahankan kemanusiaan mereka.",
                R.drawable.rm,

"https://www.imdb.com/title/tt11612120/?ref_=nv_sr_srsg_0_tt_8_nm_0_in_0_q_Sweet%2520Home"
            ),
            Film(
                3,
                "Money Heist Korea",
                "2022",
                "Pencuri menguasai gedung percetakan uang milik Korea bersatu dan menawan banyak sandera. Polisi harus menghentikan aksi mereka, serta dalang yang bersembunyi di baliknya.",
                R.drawable.duitheist, // ganti dengan nama file gambar Anda

"https://www.imdb.com/title/tt13696452/?ref_=nv_sr_srsg_0_tt_8_nm_0_in_0_q_Money%2520Heist%2520Ko"
            ),
            Film(
                4,
                "My Name",
                "2021",
                "Setelah ayahnya dibunuh, seorang wanita yang ingin membalas dendam memutuskan untuk memercayai bos kriminal yang berkuasa—dan memasuki kepolisian atas petunjuk pria itu.",
                R.drawable.ngaran, // ganti dengan nama file gambar Anda

"https://www.imdb.com/title/tt12940504/?ref_=nv_sr_srsg_3_tt_8_nm_0_in_0_q_My%2520Name"
```

```
            ),
            Film(
                5,
                "The                    8                    Show",
                "2024",
                "Indonesia's preeminent superhero and his alter ego Sancaka enter the cinematic universe to battle the wicked Pengkor and his                    orphan                    army.",
                R.drawable.delapan, // ganti dengan nama file gambar Anda
                "https://www.imdb.com/title/tt9201084/"
            )
        )
    }
}
```

Tabel 8 Source Code Soal 1 - XML

Data / Film / XML

```
package                                com.example.m3xml.data

import                                  android.os.Parcelable
import                                kotlinx.parcelize.Parcelize

@Parcelize
data                            class                            Film(
    val                            id:                             Int,
    val                           title:                          String,
    val                           year:                           String,
    val                           plot:                           String,
    val                           poster:                            Int,
    val                          imdbUrl:                          String
) : Parcelable
```

Tabel 9 Source Code Soal 1 - XML

Ui.theme / DetailFragment / XML

```
package                                com.example.m3xml.ui.theme

import                              android.annotation.SuppressLint
import                                        android.os.Bundle
import                                   android.view.LayoutInflater
import                                        android.view.View
import                                    android.view.ViewGroup
import                                 androidx.fragment.app.Fragment
import                        androidx.fragment.app.activityViewModels
import                            androidx.navigation.fragment.navArgs
import          com.example.m3xml.databinding.FragmentDetailBinding
import                        com.example.m3xml.viewmodel.FilmViewModel

class          DetailFragment          :          Fragment()          {

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

    private val viewModel: FilmViewModel by activityViewModels()
```

```
20      private val args: DetailFragmentArgs by navArgs()
21
22      override fun onCreateView(
23          inflater: LayoutInflater, container: ViewGroup?,
24          savedInstanceState: Bundle?
25      ): View {
26          _binding = FragmentDetailBinding.inflate(inflater,
27  container, false)
28          return binding.root
29      }
30
31      @SuppressLint("SetTextI18n")
32      override fun onViewCreated(view: View, savedInstanceState:
33  Bundle?) {
34          super.onViewCreated(view, savedInstanceState)
35
36          val filmId = args.filmId
37          val film = viewModel.getFilmById(filmId)
38
39          film?.let {
40              binding.ivDetailPoster.setImageResource(it.poster)
41              binding.tvDetailTitle.text = "${it.title}
42  (${it.year})"
43              binding.tvDetailPlot.text = it.plot
44          }
45      }
46
47      override fun onDestroyView() {
48          super.onDestroyView()
49          _binding = null
        }
    }
```

Tabel 10 Source Code Soal 1 - XML

Ui.theme / FilmAdapter / XML

```
1   package com.example.m3xml.ui.theme
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.view.LayoutInflater
6   import android.view.ViewGroup
7   import androidx.recyclerview.widget.RecyclerView
8   import com.example.m3xml.databinding.ListItemBinding
9   import com.example.m3xml.data.Film
10
11  class FilmAdapter(private val filmList: List<Film>) :
12      RecyclerView.Adapter<FilmAdapter.FilmViewHolder>() {
13
14      var onDetailClick: ((Int) -> Unit)? = null
15
16      inner class FilmViewHolder(val binding: ListItemBinding) :
17  RecyclerView.ViewHolder(binding.root)
18
19      override fun onCreateViewHolder(parent: ViewGroup, viewType:
```

```
Int):                    FilmViewHolder                    {
        val                      binding                   =
ListItemBinding.inflate(LayoutInflater.from(parent.context),
parent,                                                  false)
        return                          FilmViewHolder(binding)
    }

    override   fun   onBindViewHolder(holder:   FilmViewHolder,
position:                        Int)                      {
        val          film          =          filmList[position]
        with(holder.binding)                                {
            ivPoster.setImageResource(film.poster)
            tvTitle.text              =              film.title
            tvYear.text               =               film.year
            tvPlot.text               =               film.plot

            btnImdb.setOnClickListener                      {
                val    intent    =    Intent(Intent.ACTION_VIEW)
                intent.data      =      Uri.parse(film.imdbUrl)
                holder.itemView.context.startActivity(intent)
            }

            btnDetail.setOnClickListener                    {
                onDetailClick?.invoke(film.id)
            }
        }
    }

    override     fun     getItemCount()     =     filmList.size
}
```

Tabel 11 Source Code Soal 1 - XML

Ui.theme / ListFragment / XML

```
1   package                                  com.example.m3xml.ui.theme
2
3   import                                          android.os.Bundle
4   import                                 android.view.LayoutInflater
5   import                                          android.view.View
6   import                                      android.view.ViewGroup
7   import                                androidx.fragment.app.Fragment
8   import                     androidx.fragment.app.activityViewModels
9   import               androidx.navigation.fragment.findNavController
10  import               androidx.recyclerview.widget.GridLayoutManager
11  import             androidx.recyclerview.widget.LinearLayoutManager
12  import           com.example.m3xml.databinding.FragmentListBinding
13  import                   com.example.m3xml.viewmodel.FilmViewModel
14
15  class          ListFragment          :          Fragment()          {
16
17      private    var    _binding:    FragmentListBinding?    =    null
18      private      val      binding      get()      =      _binding!!
19      private      val      filmViewModel:      FilmViewModel      by
```

```
20  activityViewModels()
21
22      override                    fun                    onCreateView(
23          inflater:    LayoutInflater,    container:    ViewGroup?,
24          savedInstanceState:                                Bundle?
25      ):                            View                              {
26          _binding     =     FragmentListBinding.inflate(inflater,
27  container,                                                  false)
28          return                                           binding.root
29      }
30
31      override  fun  onViewCreated(view:  View,  savedInstanceState:
32  Bundle?)                                                          {
33          super.onViewCreated(view,             savedInstanceState)
34
35          val      filmAdapter      =      FilmAdapter(emptyList())
36
37          val  orientation  =  resources.configuration.orientation
38          if                     (orientation                      ==
39  android.content.res.Configuration.ORIENTATION_LANDSCAPE)          {
40              binding.recyclerView.layoutManager                    =
41  GridLayoutManager(context,                                       2)
42          }                            else                         {
43              binding.recyclerView.layoutManager                    =
44  LinearLayoutManager(context)
45          }
46
47          binding.recyclerView.adapter        =        filmAdapter
48
49          filmViewModel.films.observe(viewLifecycleOwner) { films ->
                val        adapter        =        FilmAdapter(films)
                binding.recyclerView.adapter          =          adapter
                adapter.onDetailClick     =     {     filmId     ->
                    val                    action                    =
    ListFragmentDirections.actionListFragmentToDetailFragment(filmId)
                    findNavController().navigate(action)
                }
            }
        }

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

Tabel 12 Source Code Soal 1 - XML

Viewmodel / FilmViewModel / XML

```
1  package com.example.m3xml.viewmodel
2
3  import androidx.lifecycle.LiveData
4  import androidx.lifecycle.MutableLiveData
5  import androidx.lifecycle.ViewModel
6  import com.example.m3xml.data.DataSource
7  import com.example.m3xml.data.Film
8
9  class FilmViewModel : ViewModel() {
10     private val _films = MutableLiveData<List<Film>>()
11     val films: LiveData<List<Film>> = _films
12
13     init {
14         _films.value = DataSource.getFilms()
15     }
16
17     fun getFilmById(id: Int): Film? {
18         return _films.value?.find { it.id == id }
19     }
20 }
```

Tabel 13 Source Code Soal 1 - XML

MainActivity.kt / XML

```
1  package com.example.m3xml
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5  import com.example.m3xml.databinding.ActivityMainBinding //
6  Ganti dengan package Anda
7
8  class MainActivity : AppCompatActivity() {
9
10     private lateinit var binding: ActivityMainBinding
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         binding = ActivityMainBinding.inflate(layoutInflater)
15         setContentView(binding.root)
16     }
17 }
```

Tabel 14 Source Code Soal 1 - XML

Layout / activity_main.xml / XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/nav_graph" />

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

Tabel 15 Source Code Soal 1 - XML

Layout / fragment_detail.xml / XML

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.theme.DetailFragment">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="16dp">

        <TextView
            android:id="@+id/tv_header_detail"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
            android:layout_marginTop="16dp"
            android:layout_marginEnd="16dp"
            android:text="Movie Detail"

android:textAppearance="@style/TextAppearance.Material3.HeadlineLarge"
            android:textStyle="bold"
```

```
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <com.google.android.material.imageview.ShapeableImageView
            android:id="@+id/iv_detail_poster"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_marginStart="16dp"
            android:layout_marginTop="16dp"
            android:layout_marginEnd="16dp"
            android:scaleType="centerCrop"
            app:layout_constraintDimensionRatio="2:3"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/tv_header_detail"
            app:shapeAppearanceOverlay="@style/roundedImageView"
            tools:srcCompat="@drawable/dukun" />

        <TextView
            android:id="@+id/tv_detail_title"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"

android:textAppearance="?attr/textAppearanceHeadlineSmall"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="@id/iv_detail_poster"

app:layout_constraintStart_toStartOf="@id/iv_detail_poster"

app:layout_constraintTop_toBottomOf="@id/iv_detail_poster"
            tools:text="Uncanny Counter Season 1 (2020)" />

        <TextView
            android:id="@+id/tv_detail_plot_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="Plot:"
            android:textAppearance="?attr/textAppearanceTitleMedium"
            android:textStyle="bold"

app:layout_constraintStart_toStartOf="@id/tv_detail_title"

app:layout_constraintTop_toBottomOf="@id/tv_detail_title" />

        <TextView
            android:id="@+id/tv_detail_plot"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
            android:textAppearance="?attr/textAppearanceBodyMedium"
            app:layout_constraintEnd_toEndOf="@id/tv_detail_title"

```

```
app:layout_constraintStart_toStartOf="@id/tv_detail_plot_label"

app:layout_constraintTop_toBottomOf="@id/tv_detail_plot_label"
            tools:text="Para Counter yang karyawan toko mi di siang hari dan pemburu iblis di malam hari, memakai berbagai kemampuan khusus untuk memburu roh-roh jahat yang mengincar manusia." />

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

Tabel 16 Source Code Soal 1 - XML

Layout / fragment_list.xml / XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.theme.ListFragment">

    <TextView
        android:id="@+id/tv_header_title"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:text="KDrama Cinema"

android:textAppearance="@style/TextAppearance.Material3.HeadlineLarge"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginTop="16dp"
        android:clipToPadding="false"
        android:paddingBottom="8dp"

app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/tv_header_title"
        tools:listitem="@layout/list_item" />

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

Tabel 17 Source Code Soal 1 - XML

Layout / list_item.xml / XML

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.material.card.MaterialCardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    app:cardCornerRadius="16dp"
    app:cardElevation="4dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">


<com.google.android.material.imageview.ShapeableImageView
            android:id="@+id/iv_poster"
            android:layout_width="100dp"
            android:layout_height="150dp"
            android:scaleType="centerCrop"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"

app:shapeAppearanceOverlay="@style/roundedImageView"
            tools:src="@tools:sample/avatars" />

        <TextView
            android:id="@+id/tv_title"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
            android:layout_marginEnd="8dp"

android:textAppearance="?attr/textAppearanceHeadline6"
            app:layout_constraintEnd_toStartOf="@+id/tv_year"
            app:layout_constraintStart_toEndOf="@id/iv_poster"
            app:layout_constraintTop_toTopOf="@id/iv_poster"
            tools:text="Judul Film Panjang Sekali" />

        <TextView
            android:id="@+id/tv_year"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?attr/textAppearanceBody2"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="@id/tv_title"
            tools:text="2025" />

        <TextView
            android:id="@+id/tv_plot_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_marginTop="8dp"
            android:text="Plot:"
            android:textStyle="bold"
            app:layout_constraintStart_toStartOf="@id/tv_title"
            app:layout_constraintTop_toBottomOf="@id/tv_title"
/>

        <TextView
            android:id="@+id/tv_plot"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_marginTop="4dp"
            android:ellipsize="end"
            android:maxLines="3"
            app:layout_constraintBottom_toTopOf="@id/btn_imdb"
            app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="@id/tv_plot_label"

app:layout_constraintTop_toBottomOf="@id/tv_plot_label"
            tools:text="Deskripsi plot film yang cukup panjang
untuk mengisi beberapa baris dan menunjukkan fungsi elipsis." />


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

style="@style/Widget.MaterialComponents.Button.OutlinedButton"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:layout_marginEnd="8dp"
            android:text="IMDB"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@id/btn_detail"

app:layout_constraintStart_toStartOf="@id/tv_plot_label"     />

        <Button
            android:id="@+id/btn_detail"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:text="Detail"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@id/btn_imdb"
/>

    </androidx.constraintlayout.widget.ConstraintLayout>

</com.google.android.material.card.MaterialCardView>
```

Tabel 18 Source Code Soal 1 - XML

Navigation / nav_graph.xml / XML

```
<?xml version="1.0" encoding="utf-8"?>
<navigation
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/listFragment">

    <fragment
        android:id="@+id/listFragment"
        android:name="com.example.m3xml.ui.theme.ListFragment"
        android:label="fragment_list"
        tools:layout="@layout/fragment_list" >
        <action

android:id="@+id/action_listFragment_to_detailFragment"
            app:destination="@id/detailFragment" />
    </fragment>
    <fragment
        android:id="@+id/detailFragment"

android:name="com.example.m3xml.ui.theme.DetailFragment"
        android:label="fragment_detail"
        tools:layout="@layout/fragment_detail" >
        <argument
            android:name="filmId"
            app:argType="integer" />
    </fragment>
</navigation>
```

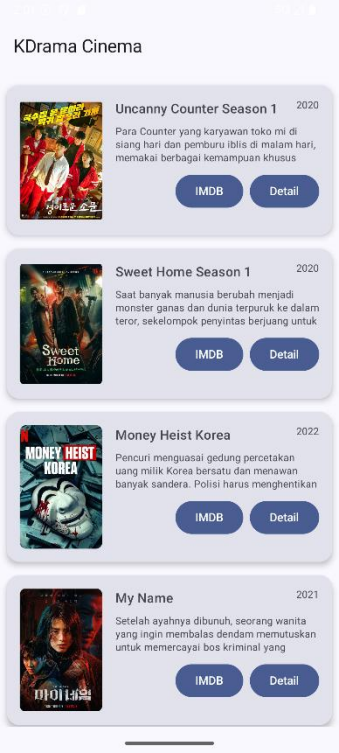
Tabel 19 Source Code Soal 1 - XML

Values/ themes.xml / XML

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="Base.Theme.M3XML"
parent="Theme.Material3.DayNight.NoActionBar">
    </style>

    <style name="Theme.M3XML" parent="Base.Theme.M3XML" />

    <style name="roundedImageView">
        <item name="cornerFamily">rounded</item>
        <item name="cornerSize">16dp</item>
    </style>

</resources>
```

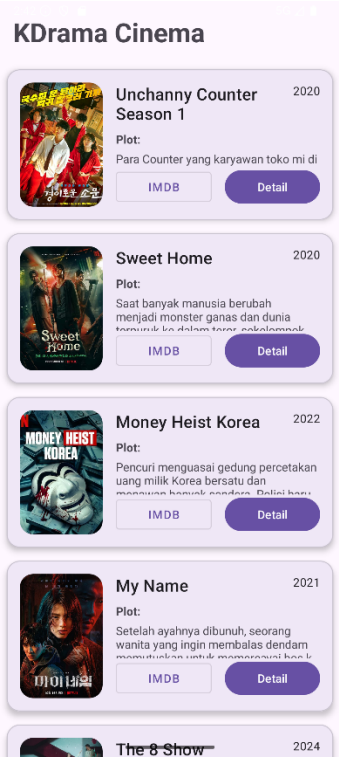
Tabel 20 Source Code Soal 1 - XML

B. Output Program



Gambar 3 Screenshot Hasil Jawaban Soal 1 - Compose

Gambar 4 Screenshot Hasil Jawaban Soal 1 - Compose



Gambar 5 Screenshot Hasil Jawaban Soal 1 - XML

Gambar 6 Screenshot Hasil Jawaban Soal 1 – XML

C. Pembahasan
Jetpack Compose

1. model/Movie.kt (Struktur Data)
File ini adalah fondasi dari data yang akan ditampilkan.

- **Pada baris 3**, data class Movie dideklarasikan. data class adalah jenis kelas di Kotlin yang tujuan utamanya hanya untuk menyimpan data. Kelas ini mendefinisikan "cetakan" atau struktur untuk sebuah objek film, yang terdiri dari id (unik), title (judul), description (deskripsi), dan imageRes (ID gambar dari drawable).

2. data/MovieRepository.kt (Sumber Data)
File ini bertindak sebagai "database" tiruan atau sumber data untuk aplikasi.

- **Pada baris 5**, object MovieRepository dideklarasikan. Penggunaan object menjadikan MovieRepository sebuah *singleton*, artinya hanya ada satu instance dari repositori ini di seluruh aplikasi. Ini memastikan data yang diakses dari layar mana pun selalu konsisten.
- **Pada baris 6**, private val movies dideklarasikan. Ini adalah sebuah List privat yang berisi semua objek Movie yang akan ditampilkan. Data film di-hardcode langsung di sini.
- **Pada baris 41**, fungsi getMovies() didefinisikan untuk mengembalikan seluruh daftar film yang ada.
- **Pada baris 44**, fungsi getMovieById(id: Int) didefinisikan untuk mencari dan mengembalikan satu objek Movie berdasarkan id yang diberikan.

3. navigation/Screen.kt (Definisi Rute Navigasi)
File ini mendefinisikan semua kemungkinan tujuan (layar) navigasi dalam aplikasi secara terstruktur.

- **Pada baris 3**, sealed class Screen dideklarasikan. sealed class digunakan di sini untuk membuat grup rute yang terbatas dan aman-tipe (*type-safe*). Ini mencegah kesalahan pengetikan nama rute.
- **Pada baris 4**, object MovieList merepresentasikan rute untuk layar daftar film.
- **Pada baris 5**, object Detail merepresentasikan rute untuk layar detail film. Rute ini didefinisikan sebagai "detail/{movieId}", di mana {movieId} adalah placeholder untuk argumen (ID film) yang akan dikirimkan.
- **Pada baris 7**, fungsi createRoute(movieId: Int) dibuat untuk memudahkan pembuatan rute lengkap ke layar detail dengan menyisipkan ID film yang sebenarnya ke dalam placeholder.

4. navigation/AppNavigation.kt (Pengatur Navigasi)
File ini adalah pusat kendali navigasi. Ia mengatur bagaimana semua layar terhubung.

- **Pada baris 11**, fungsi Composable AppNavigation dideklarasikan, yang menjadi inti dari sistem navigasi.
- **Pada baris 12**, val navController = rememberNavController() digunakan untuk membuat dan mengingat NavController, yaitu objek yang bertanggung jawab untuk mengelola navigasi (seperti Maps(), popBackStack(), dll).
- **Pada baris 13**, NavHost digunakan sebagai kontainer yang akan menampilkan tujuan (layar) Composable berdasarkan rute saat ini. startDestination diatur ke Screen.MovieList.route, yang berarti layar daftar film adalah layar pertama yang ditampilkan saat aplikasi dibuka.
- **Pada baris 17**, composable(route = Screen.MovieList.route) mendefinisikan apa yang harus ditampilkan untuk rute daftar film. Di sini, ia memanggil MovieListScreen, sambil meneruskan navController agar layar tersebut bisa melakukan navigasi.
- **Pada baris 21**, composable(route = Screen.Detail.route, ...) mendefinisikan layar detail.
- **Pada baris 22**, arguments = listOf(navArgument("movieId") { type = NavType.IntType }) mendefinisikan bahwa rute ini menerima sebuah argumen bernama "movieId" yang bertipe Int.

- **Pada baris 25**, val movieId = backStackEntry.arguments?.getInt("movieId") ?: 0 digunakan untuk mengekstrak nilai movieId yang dikirim dari layar sebelumnya.
- **Pada baris 26**, DetailScreen dipanggil dengan movieId yang telah diekstrak, memungkinkan layar detail untuk menampilkan data yang benar.

5. ui/theme/MovieListScreen.kt (Layar Daftar Film)
Ini adalah Composable untuk layar utama yang menampilkan daftar semua film.

- **Pada baris 18**, fungsi Composable MovieListScreen dideklarasikan, menerima navController sebagai parameter untuk keperluan navigasi.
- **Pada baris 19**, val movies = MovieRepository.getMovies() digunakan untuk mengambil data seluruh film dari repositori.
- **Pada baris 21**, LazyColumn digunakan untuk menampilkan daftar yang bisa di-scroll. LazyColumn sangat efisien karena hanya merender item yang terlihat di layar.
- **Pada baris 24**, items(movies) melakukan iterasi pada setiap movie di dalam daftar movies dan membuat sebuah MovieItem untuk masing-masing film.
- **Pada baris 29**, di dalam MovieItem, Card digunakan untuk membungkus setiap item daftar dengan tampilan kartu yang memiliki bayangan dan sudut membulat.
- **Pada baris 32**, modifier = Modifier.clickable { ... } membuat seluruh kartu dapat diklik.
- **Pada baris 33**, di dalam blok clickable, navController.navigate(Screen.Detail.createRoute(movie.id)) dieksekusi. Ini adalah aksi navigasi yang membawa pengguna ke layar detail, sambil mengirimkan id dari film yang diklik.
- **Pada baris 37**, Row digunakan untuk menyusun gambar dan teks (judul, deskripsi) secara horizontal.
- **Pada baris 39**, Image digunakan untuk menampilkan poster film. painterResource memuat gambar dari drawable.
- **Pada baris 45**, Column digunakan untuk menyusun judul dan deskripsi film secara vertikal.

6. ui/theme/DetailScreen.kt (Layar Detail Film)
Ini adalah Composable untuk layar yang menampilkan detail dari satu film yang dipilih.

- **Pada baris 19**, fungsi Composable DetailScreen dideklarasikan, menerima movieId dan navController sebagai parameter.
- **Pada baris 20**, val movie = MovieRepository.getMovieById(movieId) digunakan untuk mengambil data spesifik dari satu film berdasarkan movieId yang diterima dari navigasi.
- **Pada baris 22**, BackHandler(onBack = { navController.popBackStack() }) digunakan untuk menangani penekanan tombol kembali (baik fisik maupun gestur). navController.popBackStack() akan mengembalikan pengguna ke layar sebelumnya (yaitu MovieListScreen).
- **Pada baris 25**, Column digunakan untuk menyusun semua elemen UI detail secara vertikal dan diposisikan di tengah.
- **Pada baris 32**, Image digunakan untuk menampilkan poster film yang dipilih dengan ukuran yang lebih besar.
- **Pada baris 38 & 39**, Text digunakan untuk menampilkan judul dan deskripsi film dengan gaya teks yang berbeda untuk memberikan penekanan.

7. MainActivity.kt (Aktivitas Utama)
File ini adalah titik masuk aplikasi, tempat semuanya dimulai.

- **Pada baris 16**, di dalam onCreate, setContent dipanggil untuk memulai UI Jetpack Compose.
- **Pada baris 18**, Surface digunakan sebagai container latar belakang.

- **Pada baris 22**, AppNavigation() dipanggil. Ini adalah satu-satunya panggilan penting di sini, yang secara efektif menyerahkan seluruh kontrol UI dan navigasi ke Composable AppNavigation

XML

1. Data / DataSource (DataSource.kt)

File ini berfungsi sebagai sumber data statis untuk aplikasi.

- **Pada baris 5, object DataSource** digunakan untuk mendeklarasikan kelas sebagai sebuah *singleton*. Pola desain ini memastikan bahwa hanya ada satu instance dari DataSource yang digunakan di seluruh siklus hidup aplikasi, sehingga data yang diakses bersifat konsisten dan terpusat.
- **Pada baris 6, fun getFilms(): List<Film>** adalah sebuah fungsi publik yang ketika dipanggil akan mengembalikan data berupa List (daftar) yang berisi objek-objek dari kelas Film.
- **Pada baris 7, return listOf(...)** mengembalikan sebuah List yang nilainya telah ditentukan secara langsung di dalam kode (statis).
- **Pada baris 8-14, Film(1, "Unchanny Counter Season 1", ...)** merupakan proses inisialisasi atau pembuatan instance dari objek Film. Setiap argumen yang dilewatkan (seperti id, judul, tahun, plot, ID resource drawable, dan URL) mengisi properti yang telah didefinisikan di dalam data class Film.

2. Data / Film (Film.kt)

File ini mendefinisikan model atau struktur data untuk entitas film.

- **Pada baris 7, data class Film(...)** adalah sebuah fitur Kotlin untuk membuat kelas yang tujuan utamanya adalah untuk menyimpan data. Kelas ini mendefinisikan properti yang dimiliki oleh sebuah entitas film, yaitu id, title, year, plot, poster, dan imdbUrl.
- **Pada baris 6, @Parcelize** merupakan sebuah anotasi dari library Kotlin Android Extensions yang secara otomatis meng-generate implementasi dari Parcelable.
- **Pada baris 14, : Parcelable** adalah implementasi dari interface Parcelable milik Android. Mekanisme ini memungkinkan objek dari kelas Film untuk diserialisasi sehingga dapat dikirim antar komponen Android, misalnya sebagai argumen dalam navigasi antar Fragment.

3. Viewmodel / FilmViewModel (FilmViewModel.kt)

Kelas ini menerapkan pola arsitektur ViewModel yang berfungsi untuk menyimpan dan mengelola data terkait UI. ViewModel dirancang untuk bertahan dari perubahan konfigurasi, seperti rotasi layar.

- **Pada baris 11, private val _films = MutableLiveData<List<Film>>()** membuat sebuah properti privat berupa MutableLiveData. Objek ini dapat menyimpan daftar film dan nilainya dapat diubah dari dalam ViewModel.
- **Pada baris 12, val films: LiveData<List<Film>> = _films** mengekspos data _films sebagai LiveData yang bersifat *read-only*. Hal ini merupakan praktik yang direkomendasikan untuk memastikan bahwa data hanya dapat dimodifikasi oleh ViewModel, sementara komponen UI (seperti Fragment) hanya dapat mengamatinya.
- **Pada baris 14, init { ... }** adalah blok inisialisasi yang dieksekusi secara otomatis ketika instance ViewModel pertama kali dibuat. Blok ini bertugas untuk memuat data film dari DataSource ke dalam _films.
- **Pada baris 18, fun getFilmById(id: Int): Film?** adalah sebuah fungsi publik yang memungkinkan komponen lain untuk mengambil data satu film secara spesifik dari daftar berdasarkan id-nya. Tipe kembalian Film? mengindikasikan bahwa fungsi ini dapat mengembalikan null jika film dengan id tersebut tidak ditemukan.

4. Ui.theme / FilmAdapter (FilmAdapter.kt)

Kelas Adapter ini berfungsi sebagai jembatan antara sumber data (List<Film>) dengan komponen RecyclerView yang menampilkannya.

- **Pada baris 12, class FilmAdapter(...)** merupakan deklarasi kelas adapter yang menerima List<Film> pada konstruktornya sebagai sumber data untuk ditampilkan.

- **Pada baris 15, var onDetailClick: ((Int) -> Unit)?** mendeklarasikan sebuah properti fungsional (lambda) yang dapat di-override dari luar. Properti ini berfungsi sebagai *callback* untuk menangani event klik pada tombol detail, dengan mengirimkan id film sebagai parameter.
- **Pada baris 25, override fun onBindViewHolder(...)** adalah fungsi yang dipanggil oleh RecyclerView untuk menampilkan data pada posisi tertentu. Di dalam fungsi ini, data dari objek film diikat ke setiap elemen View di dalam ViewHolder.
- **Pada baris 32, btnImdb.setOnClickListener** mendaftarkan sebuah *click listener* pada tombol IMDB. Ketika tombol diklik, sebuah Intent dengan aksi ACTION_VIEW dibuat untuk membuka imdbUrl yang tersimpan pada data film.
- **Pada baris 38, btnDetail.setOnClickListener** mendaftarkan *click listener* pada tombol Detail. Ketika diklik, ia akan mengeksekusi lambda onDetailClick dan meneruskan film.id dari item yang bersangkutan.

5. Ui.theme / ListFragment (ListFragment.kt)

Fragment ini merepresentasikan layar yang menampilkan daftar film kepada pengguna.

- **Pada baris 20, private val filmViewModel: FilmViewModel by activityViewModels()** adalah cara untuk mendapatkan instance FilmViewModel yang terikat pada siklus hidup Activity. Ini memungkinkan data di ViewModel dapat dibagikan antara beberapa Fragment (ListFragment dan DetailFragment).
- **Pada baris 35, if (orientation == android.content.res.Configuration.ORIENTATION_LANDSCAPE)** adalah implementasi logika untuk layout responsif. Kode ini memeriksa orientasi perangkat saat ini dan memilih LayoutManager yang sesuai untuk RecyclerView: GridLayoutManager untuk landscape dan LinearLayoutManager untuk portrait.
- **Pada baris 43, filmViewModel.films.observe(...)** mendaftarkan Observer ke LiveData films di dalam ViewModel. Blok kode di dalam observer ini akan dieksekusi secara otomatis setiap kali data films diperbarui.
- **Pada baris 46, adapter.onDetailClick = { filmId -> ... }** mengimplementasikan *callback* onDetailClick dari FilmAdapter. Blok kode ini mendefinisikan aksi yang akan dilakukan ketika tombol detail pada sebuah item diklik.
- **Pada baris 47, val action = ListFragmentDirections.actionListFragmentToDetailFragment(filmId)** menggunakan plugin *Navigation Safe Args* untuk membuat objek aksi navigasi yang *type-safe*. Aksi ini membawa filmId sebagai argumen untuk dikirim ke DetailFragment.
- **Pada baris 48, findNavController().navigate(action)** mengeksekusi perintah navigasi untuk berpindah dari ListFragment ke DetailFragment.

6. Ui.theme / DetailFragment (DetailFragment.kt)

Fragment ini merepresentasikan layar yang menampilkan informasi rinci dari film yang telah dipilih.

- **Pada baris 20, private val args: DetailFragmentArgs by navArgs()** menggunakan delegasi properti dari *Navigation Safe Args* untuk secara aman mengekstrak argumen yang dikirimkan melalui aksi navigasi.
- **Pada baris 34, val filmId = args.filmId** mengambil nilai argumen filmId dari objek args.
- **Pada baris 35, val film = viewModel.getFilmById(filmId)** memanggil fungsi pada FilmViewModel untuk mengambil objek Film secara keseluruhan berdasarkan filmId yang telah diterima.
- **Pada baris 37, film?.let { ... }** adalah sebuah *scope function* yang melakukan pengecekan *null*. Blok kode di dalamnya hanya akan dieksekusi jika objek film tidak null, mencegah NullPointerException.
- **Pada baris 38-40**, baris-baris kode ini mengisi komponen-komponen UI (seperti ImageView dan TextView) pada layout fragment_detail.xml dengan data yang sesuai dari properti objek film (poster, judul, tahun, dan plot).

7. MainActivity.kt

File ini adalah komponen Activity utama yang berfungsi sebagai entry point aplikasi.

- **Pada baris 8, class MainActivity : AppCompatActivity()** mendeklarasikan kelas aktivitas utama yang mewarisi fungsionalitas dari AppCompatActivity.
- **Pada baris 10, private lateinit var binding: ActivityMainBinding** mendeklarasikan variabel untuk menampung instance dari kelas binding yang dihasilkan secara otomatis untuk activity_main.xml, memungkinkan akses view yang aman.
- **Pada baris 12, override fun onCreate(...)** adalah fungsi *lifecycle callback* yang dipanggil saat Activity pertama kali dibuat.
- **Pada baris 14, binding = ActivityMainBinding.inflate(layoutInflater)** melakukan *inflate* terhadap layout XML dan menginisialisasi objek binding.
- **Pada baris 15, setContentView(binding.root)** menetapkan root view dari layout yang telah di-inflate sebagai konten utama dari Activity.

8. Layout / activity_main.xml

File layout ini berfungsi sebagai kerangka dasar antarmuka pengguna aplikasi.

- **Pada baris 9, <androidx.fragment.app.FragmentContainerView ...>** adalah sebuah View khusus yang dirancang untuk menampung Fragment. Komponen ini bertindak sebagai NavHost yang akan menampilkan tujuan navigasi.
- **Pada baris 19, app:navGraph="@navigation/nav_graph"** adalah atribut yang menghubungkan FragmentContainerView ini dengan grafik navigasi yang didefinisikan dalam nav_graph.xml, yang mengatur alur perpindahan antar Fragment.

9. Layout / fragment_list.xml

File layout ini mendefinisikan struktur visual untuk ListFragment.

- **Pada baris 9, <TextView ...>** berfungsi sebagai elemen teks statis untuk menampilkan judul pada bagian atas layar.
- **Pada baris 21, <androidx.recyclerview.widget.RecyclerView ...>** adalah komponen utama pada layout ini. RecyclerView digunakan untuk menampilkan daftar data yang besar secara efisien dengan mendaur ulang View untuk setiap item.

10. Layout / fragment_detail.xml

File layout ini mendefinisikan struktur visual untuk DetailFragment.

- **Pada baris 2, <ScrollView ...>** digunakan sebagai View induk untuk memastikan bahwa konten di dalamnya dapat di-scroll jika tingginya melebihi ukuran layar.
- **Pada baris 29, <com.google.android.material.imageview.ShapeableImageView ...>** adalah komponen ImageView dari library Material Design yang mendukung pembentukan sudut, digunakan di sini untuk menampilkan poster film dengan sudut membulat.
- **Pada baris 48, <TextView android:id="@+id/tv_detail_title" ...>** dan **baris 85, <TextView android:id="@+id/tv_detail_plot" ...>** adalah komponen TextView yang masing-masing berfungsi sebagai penampung untuk menampilkan judul dan plot film.

11. Layout / list_item.xml

File layout ini berfungsi sebagai templat untuk setiap item individual di dalam RecyclerView.

- **Pada baris 2, <com.google.android.material.card.MaterialCardView ...>** membungkus setiap item dalam sebuah View berbentuk kartu, yang memberikan elevasi (efek bayangan) dan batas sudut yang jelas, sesuai dengan pedoman Material Design.
- **Pada baris 79, <Button android:id="@+id/btn_imdb" ...>** dan **baris 90, <Button android:id="@+id/btn_detail" ...>** adalah komponen Button yang menyediakan interaksi bagi pengguna pada setiap item, yaitu untuk melihat link IMDB dan untuk menavigasi ke halaman detail.

12. Navigation / nav_graph.xml

File ini mendefinisikan semua alur navigasi antar Fragment di dalam aplikasi menggunakan Navigation Component.

- **Pada baris 6, app:startDestination="@id/listFragment"** menetapkan listFragment sebagai Fragment awal yang akan ditampilkan ketika grafik navigasi ini dimuat.
- **Pada baris 14, <action ...>** mendefinisikan sebuah jalur navigasi yang valid dari listFragment (asal) ke detailFragment (tujuan).
- **Pada baris 24, <argument android:name="filmId" app:argType="integer" />** mendeklarasikan bahwa detailFragment menerima sebuah argumen bernama filmId dengan tipe data integer. Deklarasi ini memungkinkan pengiriman data antar Fragment dengan cara yang aman (type-safe).

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Adiyaus/Praktikum-PemrogramanMobile.git

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

Alasan utama mengapa `RecyclerView` masih relevan dan terus digunakan secara luas meskipun `LazyColumn` menawarkan sintaksis yang lebih ringkas adalah karena keduanya berasal dari paradigma dan ekosistem UI yang berbeda secara fundamental.

- **`RecyclerView`** adalah komponen inti dari **Android View System**, yang merupakan pendekatan tradisional berbasis XML untuk membangun UI.
- **`LazyColumn`** adalah komponen dari **Jetpack Compose**, yaitu *toolkit* UI deklaratif modern dari Google.

Meskipun `LazyColumn` lebih modern, `RecyclerView` tetap menjadi komponen yang vital karena beberapa faktor berikut:

1. **Basis Kode yang Sudah Ada (*Legacy Codebase*)** Alasan paling signifikan adalah keberadaan jutaan aplikasi di Google Play Store yang dibangun menggunakan Android View System. Proses rekayasa ulang (*refactoring*) atau penulisan ulang seluruh aplikasi hanya untuk mengganti `RecyclerView` dengan `LazyColumn` sering kali tidak efisien dari segi biaya, waktu, dan sumber daya. Oleh karena itu, dalam pemeliharaan dan pengembangan fitur pada proyek-proyek ini, pengembang tetap harus menggunakan dan menguasai `RecyclerView`.
2. **Interoperabilitas dan Proyek Hibrida** Jetpack Compose dirancang untuk dapat berinteroperasi dengan Android View System. Banyak tim pengembang yang mengadopsi Compose secara bertahap, di mana layar-layar baru dibuat menggunakan Compose, sementara layar yang sudah ada dan kompleks tetap dipertahankan dalam format XML. Dalam skenario hibrida seperti ini, `RecyclerView` dan `LazyColumn` dapat hidup berdampingan dalam satu basis kode yang sama.
3. **Kematangan, Stabilitas, dan Kontrol Tingkat Lanjut** `RecyclerView` telah diperkenalkan sejak tahun 2014 dan telah melalui berbagai iterasi optimalisasi. Komponen ini sangat matang, stabil, dan teruji dalam berbagai skenario produksi. `RecyclerView` memberikan developer kontrol yang sangat mendetail (*fine-grained control*) atas berbagai aspek, antara lain:
    - **`LayoutManager`**: Memungkinkan pembuatan struktur layout yang sangat kustom, melebihi sekadar daftar linear atau grid.
    - **`ItemAnimator`**: Memberikan kontrol penuh atas implementasi animasi untuk operasi penambahan, penghapusan, atau pembaruan item.
    - **`RecycledViewPool`**: Memfasilitasi pembagian `View` yang telah didaur ulang antar beberapa `RecyclerView` untuk mencapai tingkat optimalisasi performa yang lebih tinggi. Untuk kasus penggunaan yang sangat kompleks dan menuntut kustomisasi tingkat lanjut, `RecyclerView` masih menjadi pilihan yang solid.
4. **Ekosistem dan Basis Pengetahuan** Sebagai komponen yang telah lama ada, `RecyclerView` memiliki ekosistem yang sangat luas, mencakup pustaka pihak ketiga, artikel teknis, dan tutorial yang tak terhitung jumlahnya. Komunitas pengembang telah memecahkan berbagai

tantangan kompleks terkait `RecyclerView`, sehingga basis pengetahuan yang tersedia sangat melimpah.

5. **Keahlian dan Familiaritas Pengembang** Tidak semua tim pengembang telah sepenuhnya beralih ke Jetpack Compose. Banyak pengembang yang memiliki keahlian dan pengalaman mendalam dengan Android View System. Bagi mereka, implementasi fitur menggunakan `RecyclerView` dapat menjadi lebih cepat dan efisien karena tingkat familiaritas yang tinggi.

Kesimpulan

Meskipun `RecyclerView` memerlukan lebih banyak kode *boiler-plate* (misalnya, pembuatan `Adapter` dan `ViewHolder`) dibandingkan `LazyColumn`, ia bukanlah teknologi yang usang. Pemilihan antara kedua komponen ini tidak didasarkan pada superioritas absolut, melainkan pada **arsitektur dan teknologi dasar yang digunakan dalam sebuah proyek**.

Untuk proyek yang dibangun di atas Android View System dengan layout XML, `RecyclerView` adalah komponen standar yang tepat untuk digunakan. Sebaliknya, untuk proyek yang sepenuhnya dibangun menggunakan Jetpack Compose, `LazyColumn` adalah pilihan utama yang modern dan efisien.